

Program Development Grant Final Report Cover Sheet

DR# 125

Project Title: New module for Summer Week

Date: August 2001

Project Manager: Pat Simonson and Nancy Nelson

Section: Santa Clara Valley

Email: nancy.nelson@swe.org

Phone: _____

Deliverables: Indicate type (i.e. document, web page, brochure, etc.), title, and media (hard copy, email file, disk, etc.)

Document, Final Report, email file.

Select the one primary Strategic Priority this project addressed:

- Leadership
 Education
 Diversity
 Visibility
 Vitality

SWE Committees to which this report and deliverables would be of interest:

- Career Guidance
 Public Relations
 Multi Cultural Committee
 Continuing Devel.
 Publications
 Other: _____
 Membership

Project Audience (age, sex, diversity) Female: XXX Male:

<i>Age Group</i>	<i>No. Actual/Proposed</i>	<i>Diversity</i>	<i>No. Actual/Proposed</i>
• Elementary	/	• Caucasian	/
• Middle School	/	X African American	/
X High School	<u>20/20</u>	X Hispanic	/
• College	/	X American Indian	/
• Professional	/	• Pacific Islander	/
• Other _____	/	• Asian	/
Contact Hrs: _____	per_attendee	• Other _____	/

SWE Volunteers (No.) 30

Estimated Total Hours: 90

Non-SWE Volunteers (No.) 30

Estimated Total Hours: 40

Amount of Grant: \$4320 **Total Final Expenses:** \$3780 **Amount SWE Owes You:** _____

-or-

Amount You Owe SWE: \$540

Executive Summary: A short summary of what the Project was and what it accomplished.

The Microsoft Corporation Equal Access Grant allowed SWE-SCV to develop and present a new java computer programming module for the Summer Week program of engineering workshops. The new module allowed junior and senior high school girls of underrepresented ethnic groups to experience creating a software program in a university environment with women engineers as mentors. This program was part of the Get Science, Engineering, and Technology (GetSET) program of SWE-SCV. Details are in the associated document.

The final report should contain the following information:

- Final Report Cover Sheet
- Recognition of SWE and the Donor
- Narrative Description, including program goals, schedule, program conduct and content.
- Measured results (number and mix of attendees, surveys, other measures of impact, etc.) including comparisons to prior data or prior expectations, what constitutes success, use charts where appropriate.
- Publicity or other public attention. (Attach copies of press clippings, list of dates and stations of TV radio publicity with short description.)
- Description of deliverables. If not included in this file, describe format, title, etc. (i.e. a video titled "xyz" or a series of web pages at <http://www....>)
- Lessons learned, problems encountered and future plans (what you would do differently if you did this again.) If the project design you followed differed significantly from the original plan describe the nature of and the reasons for the changes.
- , include all funding sources for income and expenses and show amount paid by the Grant and the balance due.
- Appendices:
 - Deliverables: Deliverables can be sent by mail to the committee chair separately, but must be clearly marked "DR# XXX ATTACHMENT ABC". It is preferred that deliverables be presented in a format that is "web ready" to facilitate sharing of the information with other SWE sections who may want to replicate the projects. However, electronic WORD or Text files are also acceptable.
 - Press Releases
 - Publicity Received, e.g. newspaper articles, newsletter announcements.

Final Report
Microsoft Corporation Equal Access Grant
Society of Women Engineers, Santa Clara Valley section
Get SET (Science, Engineering, Technology)
November 30, 2001

Description

The Society of Women Engineers, Santa Clara Valley (SWE-SCV) section successfully completed the 2001 Summer Week portion of the on-going Get Science, Engineering, and Technology (Get SET) program. Get SET is a program for high school girls in ethnic groups that are underrepresented in math and science careers. This includes Latinas, African Americans, Pacific Islanders, and American Indians. The goal is to encourage these girls to finish high school, attend college, and to expose them to opportunities in math, science, and engineering. These girls are in adolescence, which is typically a time when girls have difficulty expressing their identity and lose the confidence they had when they were younger. In addition, these girls are constantly exposed to influences at school, at home, and among their peers that discourage them from taking an interest in math and science. The Get SET program teaches girls about the application of math and science to real world situations and encourages them to consider math, science, and engineering careers.

One component of Get SET is an intense one-week, hands-on science and math camp where girls work in small teams with professional women engineers and scientists to learn engineering and science concepts. This year, the camp was held at Santa Clara University, where the girls stayed in dorms, attended workshops, and were exposed to a university setting. They began their week on Sunday evening, worked and played hard all week. It ended with a banquet on Friday that they themselves hosted to commemorate the conclusion of the year and to thank the adult volunteers of the week.

The Microsoft Corporation Equal Access Grant allowed SWE-SCV to develop and present a new java computer programming module. This workshop is based on the book *Gentle Introduction to the Art of Object Oriented Programming* written by J. Bergun, M. Stehlik, J. Roberts, and R. Pattis, published by Wiley & Sons in 1997.

The java class was held for a full day and covered the following topics:

- Creating animated pictures or applets for web pages
- The “nuts and bolts” of the java language
- Basic Java programming

The girls learned a little about programming and had hands-on opportunity to create their own small application.

Measured Results

Fifteen seniors and five juniors attended the new programming module of the Get SET program this year. Thirteen of them submitted course evaluations after the workshop. The results are as follow:

- 12 out of 13 surveys returned enthusiastically endorsed the java programming course and would recommend the course to others,
- 9 of the 13 rated the course as “very interesting”.
- Over 50% of the girls commented that they learned more about their team working skills in this workshop

Some of the comments from the girls about what they learned were:

“I’ve learned that at times I have patience especially when I’m trying to complete a project I enjoy doing.”

“That I would have to be more careful next time so I don’t lose any details. I can do anything as long as I put my mind in it.”

“That I could learn something in a short amount of time and see the fruits of my labors.”

The graduation status and post-graduation plans of these girls will not be known until next summer or later, but SWE will be collecting this data. We believe this workshop has helped these girls towards better computer understanding and enriched their choices for education and their career.

Publicity

The workshop was advertised to the Get SET participants and their parents, to the 100 plus adult volunteers in various local technology companies, and in the SWE-SCV newsletter with a readership of over 300. It was also advertised on the Santa Clara University campus, and to the corporate donors: Adaptec, Applied Materials, Inc., Lockheed Martin Missiles & Space, Lockheed Martin Technical Operations, and United Defense.

Deliverables

The java workshop materials consisted of four documents explaining java and java programming, one power point presentation, one sample web page, and seven java source code files. These materials have been sent separately..

Lessons Learned

The workshop will be modified slightly next year. We will take into account the student’s evaluations and feedback. One improvement will be to provide more detailed information that the girls can take home to continue practice on their own.

Funding Details

The Equal Access Grant for GetSET was \$4,700. The funds were spent in the following ways:

Daily cost of room and board per girl \$115.00 for 20 girls	\$2,300
Tshirts for girls and volunteers	\$ 680
Software \$80.00 for 10 copies that were shared	\$ 800
TOTAL	\$3,780
Amount of Grant	\$4,320
Amount to be returned	\$ 540

Intro to Java Programming

Instructor's Manual

Get SET Summer Week 2001

Report by Gwen Byard, gwen.byard@home.com
November 4, 2001

Overview:

This workshop gives an introduction to the Java programming language. It was first taught on August 2, 2001 to a class of approximately 15 high school seniors as a part of the Society of Women Engineers' Get SET program. These young women had varied backgrounds and experience with programming. For many of them, it was their first time writing a computer program. At all times there were at least three adult instructors, who could give assistance as needed. A ratio of at least one instructor per five girls is recommended. The instructors all had experience writing computer programs, although not all had experience with Java.

In the morning, the instructors introduced themselves and then gave a brief overview presentation on programming languages, focusing on Java in particular. After this introduction the girls learned how to make banners and buttons using a graphical interface tool called JavaRazor. Following this activity they wrote seven programs in Java using CodeWarrior. Through this exercise they learned the basics of programming and the Java language. The workshop ran from 9 a.m. to 4 p.m., with a few short breaks and an hour break for lunch.

Software and Hardware Required:

Hardware:

- Personal Computers – either one per student or one for every two. The computers must support:
 - Windows 95 or higher Operating System.
 - CD-ROM drive for installation.
 - 64MB RAM, 150 MB available hard disk space.
 - Floppy Disk drive preferable
- Diskettes – one per student (if Floppy Disk drive is available). These diskettes contained the Java programming templates and a blank webpage for their applets. If a Floppy Disk drive is not available, the files will have to be pre-loaded on the machines or downloadable.
- Handouts (see Appendices)
- Instructor's PC connected to projector for demonstrations
- Projector screen for viewing
- Internet connection (not necessary, but helpful)

Software:

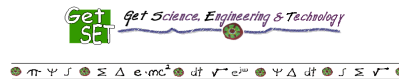
- Ulead JavaRazor Version 1.0 (or later) by Ulead Systems
- Code Warrior Learning Edition for C/C++/Java (only install Java portion) version 2.0 or higher by MetroWerks.
- Netscape or Internet Explorer for viewing Java Creations
- Notepad or other Text Editor
- Microsoft PowerPoint for presentations

Suggested Schedule:

- 9:00 – 9:20: Introductions of instructors, Java presentation, overview, and demo
9:20 – 10:45: Java Applet creation: Making Banners and Buttons
10:45 – 10:55: Break
10:55 – 12:00: Programming in Java overview, step through the first program with the students, do second program.
12:00 – 1:00: Lunch Break
1:00 – 3:50: Do programs three through seven. If the students finish early, they can go back and work some more on their Applets. Take a ten-minute break at some point when needed. Another good break is to have students volunteer to show their applets to the class.
3:50 – 4:00: Wrap up and clean up.

Introduction:

We started the workshop by having the instructors introduce themselves and talk a little about their jobs and other interests. Then one of the instructors gave a brief overview of what programming languages are and why Java is unique. Below are the power point files used – these slides and the accompanying notes are available in a separate file, java01.ppt.



Get SET Summer Week '01
Intro to Java Class
August 2, 2001



We started off by asking the students: “what is a programming language?” Then we asked them about Java and what it is used for.

Introduction

• π • ψ • \int • Σ • Δ • $e=mc^2$ • dt • $\sqrt{-c^2}$ • Ψ • Δ • dt • \int • Σ • $\sqrt{-c^2}$ •

What is a computer programming language?

What is Java?

What is it used for?



A few points that we mentioned regarding Java:

- It is a portable programming language – it can be run on any computer that has Java support on it. This isn't true of every programming language.
- It is a safe programming language – it prevents faulty programs from crashing your computer. Therefore, it is very safe to download Java programs from the Internet to your computer.
- It is used for many things, including graphical computer programs and programs that have to run across networks such as the Internet.

We did not talk for very long because typically it has been more effective in these types of workshops to limit the introduction time and get the students started in the “hands-on” activities as soon as possible.

Java Applet Creation: Making Banners and Buttons:

For this activity we started by talking about what is a Java Applet and pointed to some examples on the Internet:

Morning Activity – Create Java Applets

• π • ψ • \int • Σ • Δ • $e=mc^2$ • dt • $\sqrt{-c^2}$ • Ψ • Δ • dt • \int • Σ • $\sqrt{-c^2}$ •

What is a Java Applet?

<http://javaboutique.internet.com/ABCMenuMan/>

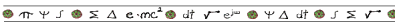
<http://javaboutique.internet.com/TOV/>

<http://javaboutique.internet.com/billsClock/>



The point we wanted to get across is that an Applet is a graphical computer program contained in a web page. The first link on the Power Point slide above is a menu with buttons that highlight when you run the mouse over them. The second link is a three-D object that you can rotate in different directions. The third link is a real-time clock. These links should be verified before the day of the workshop to make sure they are current.

Morning Activity – Create Java Applets



Demo of:

Ulead Animation Applet

Ulead Button Applet



Then we showed the students the JavaRazor application on the instructor’s PC and gave them a demo of the application and the process you go through to make an applet. Then the girls started making their own applets, and the instructors helped them as needed. For instructions on how to use the JavaRazor application, see **Appendix A**.

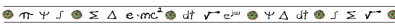
Programming in Java:

There were seven tasks in this part of the workshop.

Task 1:

We began with the first task, “Printing to the screen.” After explaining the purpose of the task, we took the girls through it step by step: how to open the CodeWarrior software, edit a source file, compile it and run it.

Task 1: Printing to the screen



Open “sourcefile1.java” using Notepad
what words do you see?

Use the “system.out.println()” command to
print a statement

Demo of first task



See **Appendix B** for how to run Code Warrior, and **Appendix C** for the descriptions of the tasks and hints.

See **Appendix D** for the skeleton Java files the students were given on the diskettes. This Appendix also has suggested solutions. We put the suggested solutions on the overhead projector because in most cases the students didn’t even know where to start. Just typing in the correct keywords and getting the programs to compile and run correctly was challenging.

Before stepping through the first program, we talked about what the different keywords meant:

class – part of a Java program – you can have many classes in a Java program, but for our purposes we are writing programs with only one class.

main – when you run a Java program, it always starts with the part of the program called main, this is the starting point for the program.

sourcefile1 – this is the name of the class in this case

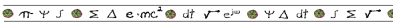
// - any words next to a double slash are comments. The computer will ignore comments.

{ } – these curly braces are used to start and finish a command in the Java programming language.

Task 2:

This task taught the students about variables and some elementary computer math. They had to print out the result for some basic math equations.

Task 2: Variables and Computer Math



```
int result;  
result = 3;  
result = 4*3;  
system.out.println(result);
```

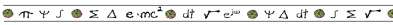


Before starting, we explained that a variable stores a value – in this class, we will use integers only. You can assign a variable using an equals sign. See Appendices C and D for more details.

Task 3:

This task showed the students how to interact with a computer program as it is running. When one types in their name, it will print out “Hello, *name*.”

Task 3: Interactive Programming



What is an example of an interactive program?
argv[0], argv[1]...

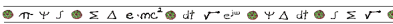


Before starting, we asked students for examples of interactive programs. One example is a password program that takes a user's name and password.

Task 4:

We started task 4 by asking the students: "When does a computer program need to make a decision?" If we go back to the password example, a computer needs to decide whether or not to let someone log in to the system based on the password they type.

Task 4: If Statements



When does a computer program need to make a decision?
if (result == 2)
system.out.println("result is 2!");
else
system.out.println("result is not 2!");

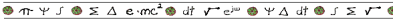


For this task, the students made an elementary password program and then tested it by entering both correct and incorrect passwords.

Task 5:

We then explained what a For Loop is and asked the students why they think it is useful. One reason is you don't have to repeat a task over and over; you can have the computer do it for you.

Task 5: For Loops



Why do you use for loops?

```
int i;
for (i=0; i<20; i++)
{
    System.out.println("hello world.");
}
```

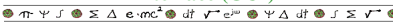


For this task, the students wrote a program to print out the numbers 1 through 10 using a For Loop.

Task 6:

This task introduced the students to the process of creating a GUI. We first asked them what a GUI is and what it is used for.

Task 6: Print out a Graphical User Interface (GUI)



What is a GUI used for?

Size the window

Name the window

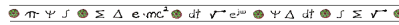


Then the students created a GUI using the template given for Task 6. They created a window, and they were able to size it and name it.

Task 7:

At the end we let the students experiment by making their own programs, using the commands they had learned. They then showed their creations to each other and the instructors.

Task 7: Create Your Own Program



Use the commands you've learned to create your own program!



Some of the students finished this task before the end of the class. We gave them a Java programming book, which allowed them to try a few additional programs. For future classes, we would recommend preparing a few optional exercises for the students who finish faster than the others.

Feedback and Possible Improvements:

Overall the workshop was well received by the students. There was a range in how interested the students were and how fast they were able to complete the programming tasks. Several instructors suggested changing the seventh program to be a more structured program that tied all of the previous six assignments together. We would also recommend preparing a few optional exercises for the quicker students who finish before the end of the workshop.

Having enough instructors is essential – the students will have many questions and some need a lot of hand-holding. Also, it was good to put up the solution on the overhead projector, at least for the first few tasks, so the students could refer to it. Many of the students didn't even know where to start, since it was their first time ever programming a computer. We also waited until everyone was done with each task before moving on to the next one. We allowed the quicker students to start the next task on their own, but we would not do the presentation and explanation of the next task until everyone was done.

The handouts were helpful too (see appendices) so the instructors didn't have to repeat themselves when explaining basic tasks. We gave out the "Computer Language Overview" (**Appendix E**) at the end so interested students could learn more about Java and computer languages. It would have taken too much time to go over this handout in detail during the class.

Appendix A – How to Make an Animated Applet:

1. Open the Ulead Animation Applet program. Click the **New** button in the **File** menu to begin a new project. You will see a box with grid lines, which is the background for your animation.
2. In the Properties Pane on the upper right of the screen click the **General** tab. Enter in the Applet's maximum **width** and **height**. The dimensions are in pixels.
3. Next, select the **Play Control** tab and, if you want, select the Infinite Loop option to enable the animation to play endlessly on the webpage. For example, if you only want to play it 10 times, click on the infinite loop option so that the check mark is not in the box and then type in the number 10.
4. On the **Background** tab, select to fill the applet with either a color, a gradient or an image. You can pick an image from the **image** folder.
5. Select the **Border** tab and set a border around your applet, if necessary. Next, click the **Advanced** tab and enter in the horizontal and vertical shifting you want to apply to it - this lets you position the applet anywhere within a web browser page. Finally, enter the **Status Bar** text that you want to appear whenever a user moves their mouse over the applet.
6. Select a Sprite from the **Sprites** menu. Sprites are special animated components for each animated applet. Each sprite has its own properties and effects associated with it – explore the different possibilities. If you want to learn about a particular Sprite, click on its help button. To add a new sprite to your animation, select one from the Sprites menu and then customize it. There are image files you can use in the **Image** folder. Once you finish setting up your sprite, click OK. The Sprite appears in the Workspace. You can position different sprites in the workspace by moving them with your mouse.
7. On the **Standard** toolbar, click the **Timeline** button (the clock). Now you can select the sprite and adjust its duration and its start and end point within the applet.
8. After you've added all your Sprites, click the **File: Output HTML** menu command. This saves your work as an HTML file with all of the corresponding Java classes necessary for properly displaying it. Save it in the same directory as your webpage (the floppy disk).
9. To put the Applet into your web page, open both the applet HTML file and your webpage in **Notepad** and cut the code from the applet to the webpage and then save. Make sure you copy the part of the code that is in the “<applet>” markers and put it in the “<body>” section of the webpage. When you load your new web page, the Java animation should load.

How to make Buttons:

1. Start Ulead Button Applet. This tool creates buttons on Web pages. The buttons that you make with Button Applet can move, glow, and change according to the movement of a mouse. To start a new button, click on **File** and **New**.
2. Pick the number of buttons you want. You can add buttons by clicking on **Edit** and **Insert Button**.
3. Define the basic size and color of your buttons using the General and Advanced tabs.
4. You can then adjust and modify each individual button by clicking on each button you want to edit. Change the shape, color, and text. Import **Image** files to be used as patterns, customize the way the buttons appear when clicked on, or adjust the effects of the buttons when the user's mouse passes over them. Make the text glow with neon, apply shadows, gradients, or change the text color. You can link your button to a webpage by putting a web address in the **URL** box under the **General** tab.
5. When you're ready to save your work, click the **File: Output HTML** menu command. This saves your work as an HTML file with all of the corresponding Java classes necessary for properly displaying it. Save it in the same directory as your webpage (the floppy disk).
6. To put the buttons into your web page, open both the applet HTML file and your webpage in **Notepad** and cut the code from the applet, paste it to the webpage and then save. Make sure you copy the part of the code that is in the "<applet>" markers and put it in the "<body>" section of the webpage. When you load your new web page, the Java buttons should load.

Appendix B – Create a Java Application:

Follow the steps below to create a Java program.

1. Launch the CodeWarrior Integrated Design Environment (IDE) Program.

To do this, click the **Start** button and select **CodeWarrior IDE** from the **CodeWarrior for Windows, Learning Edition, v2** group. This runs the program `IDE.exe`

2. Create a project using CodeWarrior stationery.

Projects hold all the files needed to run a program. Stationery is a template for creating a project. Choose **File >New**. The **New** window appears with a list of options in the **Project** tab.

a. Select the project stationery type.

Select **Java Stationery** from the **Project** pane on the left side of the window.

b. Name the new project file.

In the **Project name** edit field, type `myproject.mcp`. The extension `.mcp` is the standard extension used by Metrowerks (the maker of CodeWarrior) for CodeWarrior project files.

c. Specify a location for the project.

Click **set...** (next to the **Location** text field) and use the standard file dialog to specify the location for the new project. (put this on your floppy disk) Click **OK** to continue. The **New Project** window is displayed.

d. Select a project stationery.

Expand the **JDK 1.2** group. Choose the **Java Application** project stationery.

Click **OK** to continue. The CodeWarrior IDE creates a project with all the required files and the project window is displayed.

3. Remove the placeholder source file from the project.

A source file contains instructions written in Java for the computer to execute. A placeholder source file, `TrivialApplication.java`, exists in the new project that you just created. We want to remove this file because it's a "dummy" file not needed for this project.

To remove the file from the project, open the **Sources** group in the Project window, select the file `TrivialApplication.java`, then choose **Edit >Delete** and click **OK** in the next dialog box.

4. Open a template source file.

Choose **File >Open**. Open a source file from your floppy disk. Start with source file 1.

5. Edit the source file.

Edit the text as you would in a word processing application to create your program. Refer to the instructions for each template.

6. Save the source file.

To save the new source file, choose **File >Save**. The ".java" extension enables the CodeWarrior IDE to recognize this file as a Java source code file.

7. Add the file to your project.

With the editor window front-most, choose **Project >Add sourcefilex.java to Project**, where "x" is the number of the source file. The IDE displays a window asking to which build targets the file should be added. Ensure that both build targets are selected, and click **OK**. The file sourcefilex.java is added to the project window.

8. Specify the Main Class name.

The main class is the class the Java program will execute first. By replacing the file TrivialApplication.java we have changed the name of the main class from TrivialApplication to sourcefilex. The Target Settings should reflect this change.

Choose **Edit >Java Application Release Settings...** to display the **Target Settings** window. Click **Java Target** on the left side of the Target Settings window to display the Java Target settings panel. Change the **Main Class** to sourcefilex. Click **Apply** to save your changes. Close the Target Settings window.

9. Compile, link, and run the example.

Choose **Project >Run**. The project is compiled, linked, and run. For the first program, you should see a console window with the text "Hello World!!!" printed on the first line.

10. The next time you create a program, you don't have to create a new project.

For the purpose of this workshop, use the same project each time you write a program. Just follow **step 3** and delete the current source file from the project, close the window, then repeat **steps 4-9**. In **step 8** be sure to change the main class to the new source file.

Appendix C – Java Programming Template Instructions:

Getting started with the source files:

There are seven different source files on your floppy disk. Click on **Notepad** to open the program for editing, and then open **sourcefile1.java**. You will see the following statements:

```
public class sourcefile1
{
    public static void main(String[] argv)
    {
        // print the statement "hello world!"
    }
}
```

These statements are a part of every program we will write today. Here are explanations of a few of the key words:

class – part of a Java program – you can have many classes in a Java program, but for our purposes we are writing programs with only one class.

main – when you run a Java program, it always starts with the part of the program called main, this is the starting point for the program.

sourcefile1 – this is the name of the class in this case

// - any words next to a double slash are comments. The computer will ignore comments. Therefore, the statement: ‘print the statement "hello world!"’ will be ignored.

{ } – these brackets are used to start and finish a command in the Java programming language.

Task 1 – printing to the screen:

In this task, you will write a computer program to print the words, “Hello world!” to the screen. To do this, open sourcefile1.java using the text editor. In the space where the comment says to insert the print statement, enter the following print command:

```
System.out.println(“put any text here”);
```

Task: Use sourcefile1.java to print the words “Hello world!” to the screen

Once you edit your program, save it and then compile and run it. Compiling the program translates your commands into a language readable by the computer. **Follow the instructions in the “Create a Java Program” handout to compile and run your program. Show the results to a workshop leader.**

Task 2 – Variable declarations and mathematical functions:

A **variable** can hold many values, the same way as they do in algebra. Variables are used in computer programs to store numbers, letters and words, and these values can change as the program executes.

To create a variable, you **declare** it. For the purposes of this class, we will make all variables **integers**, or whole numbers without fractions. To declare the variable “result” you do the following:

```
int result;
```

One way to assign a value to a variable is to use an equals sign. To set **result** equal to 3, do the following:

```
result = 3;
```

To do mathematical functions, use the signs “+”, “-“, “*” and “/” for add, subtract, multiply and divide, respectively. To set result equal to 9 times 9, do the following:

```
result = 9*9;
```

To print result to the screen, use the **System.out.println** command as follows:

```
System.out.println(result);
```

Task: Use sourcefile2.java. Print out the results of 2 times 2, 4 plus 4, 8 minus 8. Show a workshop leader when you have finished.

Task 3 – Interactive programming:

Sometimes it is useful to have a program respond to the input that the user gives it. For example, if you login to an email account, you give the program your user name and password, and it checks to see if you are a legitimate user. In Java, the variable “**argv**” stores the input that a user gives. The first input is “**argv[0]**”, the second input is “**argv[1]**”, the third is “**argv[2]**”, etc. This variable can be checked and printed.

Task: Print out “Hello, <name>” when a user types in a name. Use the “System.out.println” statement that you learned earlier. If you mix a variable (i.e. argv) with text in the same “System.out.println” statement, you must separate them using a “+”.

NOTE: Be sure to do this step!!

Choose **Edit >Java Application Release Settings...**to display the Target Settings window. Click **Runtime Settings** on the left side of the Target Settings window to display the **Runtime Settings** panel. In the **Program Arguments** Field, type in the name of the person you want to say Hello to.

Task 4 – Using “If” statements:

Often a computer program needs to make a decision. For example, it could take two different actions, depending on the result of a mathematical equation. One way to describe these options is using an **if statement**. Below is an example:

```
if (result == 2)
    System.out.println(“result is 2!”);
else
    System.out.println(“result is not 2!”);
```

If result equals 2, the computer prints out “result is 2!”, otherwise it prints “result is not 2!.” The “==” sign is used to compare numbers. If you compare words that are input by the user, you should use “**argv[0].equals(<value>)**.”

Task: Make a password program sourcefile4.java. When running your program, if a user types in the correct password (make up a password) print out “login successful.” If they type in the wrong password, print out “login incorrect.” Be sure to use the command “argv[0].equals(<value>)” to compare. Show this program to a workshop leader before moving ahead.

NOTE: Be sure to do this step!! (similar as task 3)

Choose **Edit >Java Application Release Settings...**to display the Target Settings window. Click **Runtime Settings** on the left side of the Target Settings window to display the **Runtime Settings** panel. In the **Program Arguments** Field, type in the password you would like to try. Try a correct and then an incorrect password.

Task 5 – For Loops:

What would you do if you wanted to print out “hello world” 20 times? You could write the “**System.out.println**” command 20 times, or you could use a **for loop** to avoid this repetition. Here is a for loop for printing out “hello world” 20 times:

```
int i;
for (i=0; i<20; i++)
{
    System.out.println(“hello world.”);
}
```

In this example, “i” is a variable that is used as a counter. It starts at 0, and continues while it is less than 20. “i++” is a command that increments the value of i by one each time the loop is executed.

Task: Use sourcefile5.java to create a for loop that prints out the numbers 1 through 10. Show this to a workshop leader when finished.

Task 6 – Print out a Graphical User Interface (GUI):

Task: Use the code in sourcefile6.java as a starting point to print out a graphical window (GUI). Name your GUI in the quotes indicated, and size the GUI by number of pixels. Run the program several times with different sizes of the window.

This program uses the following commands:

```
javax.swing.JFrame myFrame = new javax.swing.JFrame(“Name your GUI”); -  
used to name the GUI  
myFrame.setSize(set the size of your GUI: width, height); - to set the size of the GUI  
myFrame.setVisible(true); - to make the GUI visible
```

Task 7 – Create your own program:

Now is the time to take what you’ve learned and create your own program. Use sourcefile7.java as a starting point. Be creative and think of a good way to use the Java language. At the end of the day you will show your program to the workshop leaders and the other students in the class.

Appendix D – Programming Templates and Suggested Solutions

Task 1 template:

```
public class sourcefile1
{
    public static void main(String[] argv)
    {
        // print the statement "hello world!"
    }
}
```

Task 1 suggested solution:

```
public class sourcefile1
{
    public static void main(String[] argv)
    {
        // print the statement "hello world!"
        System.out.println("hello world!");
    }
}
```

Task 2 template:

```
public class sourcefile2
{
    public static void main(String[] args)
    {
        //make your variable declaration(s) here (int)

        //print out the results of 2 times 2, 4 plus 4, 8 minus 8
        //use *, +, -
    }
}
```

Task 2 suggested solution:

```
public class sourcefile2
{
    public static void main(String[] args)
    {
        //make your variable declaration(s) here (int)
        int result;
        //print out the results of 2 times 2, 4 plus 4, 8 minus 8
        //use *, +, -
        result = 2 * 2;
        System.out.println(result);
        result = 4 + 4;
```

```
        System.out.println(result);
        result = 8 - 8;
        System.out.println(result);
    }
}
```

Task 3 template:

```
public class sourcefile3
{
    public static void main(String[] argv)
    {
        // put the print statements in here that will say
        // "Hello, <name>" when a person types in their name
    }
}
```

Task 3 suggested solution:

```
public class sourcefile3
{
    public static void main(String[] argv)
    {
        // put the print statements in here that will say
        // "Hello, <name>" when a person types in their name

        System.out.println("Hello "+ argv[0]);
    }
}
```

Task 4 template:

```
public class sourcefile4
{
    public static void main(String[] argv)
    {
        //put if statement here
        //make up a password. If the user types in the password, print
        //"login successful." If the wrong password is typed in print
        //"login incorrect."
        //include println statements as needed
    }
}
```

Task 4 suggested solution:

```
public class sourcefile4
{
```

```

public static void main(String[] argv)
{
    //put if statement here
    //make up a password. If the user types in the password, print
    //"login successful." If the wrong password is typed in print
    //"login incorrect."
    //include println statements as needed
    if (argv[0].equals("mypassword"))
        System.out.println("correct password, login complete");
    else
        System.out.println("incorrect password, login failed");

}
}

```

Task 5 template:

```

public class sourcefile5
{
    public static void main(String[] args)
    {
        //use a for loop to print out the numbers 1 through 10
    }
}

```

Task 5 suggested solution:

```

public class sourcefile5
{
    public static void main(String[] args)
    {
        //use a for loop to print out the numbers 1 through 10
        int i;
        for (i=0; i<10; i++)
            System.out.println(i+1);
    }
}

```

Task 6 template:

//This program prints out a window. Set the size of your GUI and
//name it

```

public class sourcefile6
{
    public static void main(String[] argv)
    {

```

```
    javax.swing.JFrame myFrame = new javax.swing.JFrame("Name your GUI");
    myFrame.setSize(set the size of your GUI: width, height);
    myFrame.setVisible(true);
}
}
```

Task 6 suggested solution:

//This program prints out a window. Set the size of your GUI and
//name it

```
public class sourcefile6
{
    public static void main(String[] argv)
    {
        javax.swing.JFrame myFrame = new javax.swing.JFrame("My GUI");
        myFrame.setSize(300, 500);
        myFrame.setVisible(true);
    }
}
```

Task 7 template:

```
public class sourcefile7
{
    public static void main(String[] argv)
    {
        // make your own program!
    }
}
```

Appendix E – Computer Language Overview:

Questions about Computer Languages & Computer Programs

What is a language?

We all have an idea in the back of our minds about what language is, but let's look at some formal definitions. According to Webster's New Collegiate dictionary, language is:

1a) the words, their pronunciation, and the methods of combining them used and understood by a considerable community.

2) a systematic means of communicating ideas or feelings by the use of conventionalized signs, sounds, gestures, or marks having understood meanings.

Language can be written like this text, spoken aloud, gestured like sign-language. Typically, one person will use language to communicate with another person.

What are “computer” languages?

Computer languages are a special kind of language. People use computer language to communicate with computers. People use computer languages to communicate with computers. The word "command" is often used because the computer is being told what tasks to do. Because of the way computers work, commands have to be very precise. Computers do what you tell them, not what you think you tell them.

Who uses computer languages?

Everybody who uses a computer is using computer languages indirectly. For example, when we use Netscape Communicator, we are telling our computer to go to another computer and ask it for the information (the website) that we want and then tell us what it found. When we use Microsoft Word to write a poem, we're telling our computer to save the words that we're typing and even to make sure that we spelled them correctly. There can be more than one computer language involved, depending on the programs and the computers.

Computer programs are collections of instructions in a computer language that are saved to be used over and over again.

Who writes programs?

Many people with different interests and backgrounds write computer programs. Anyone who can understand computer language and has something that she wants a computer to do will write a Program. Scientists like marine biologists or medical researchers sometimes write computer programs to study endangered-species data or analyze drug research. Some business people write programs to help sell products on the Internet. Sometimes people write programs for computers that that we wouldn't think are being used. For instance, someone wrote a program in computer language to time traffic signals. Programs are running wherever there are computers.

Often people will form a team to make more complicated programs. Eventually, whole companies are formed to work on really complicated programs. Quite frequently, companies will take these saved programs and sell them to other companies or to consumers like us. Then, we can buy programs and use them without having to rewrite all the instructions to tell the computer what to do.

Why use the Java language?

Many different varieties of computer languages exist. Some are old and not used a lot today. Some are fairly new and common. Certain computer languages can be used more easily to write certain types of programs. For example, the person writing a program to control traffic lights might use a different computer language than someone writing Microsoft Word.

Java, although it is not necessary, is often chosen for making the following types of programs:

- Programs that run in a Web Browser
- Programs that have to run on more than one type of computer
(big servers, or little personal computers)
- Programs that have to talk to other programs over to more than one computer
(e.g. Programs that talk over the Internet or the Phone)
- Programs that have graphical or user-friendly interface
(e.g. Programs that use a mouse to communicate instead of keyboard only)

A person writing one of these types of programs might chose Java because it is easier to write these instructions in Java than in another computer language.

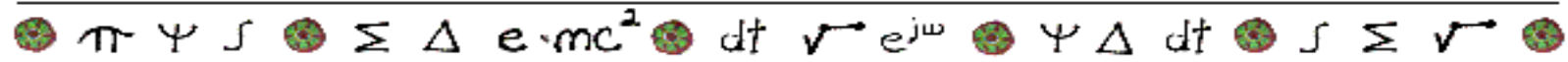


Get SET Summer Week '01

Intro to Java Class

August 2, 2001

Introduction

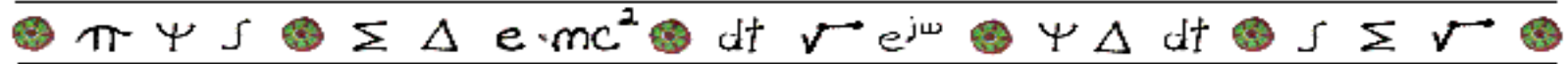


What is a computer programming language?

What is Java?

What is it used for?

Morning Activity – Create Java Applets



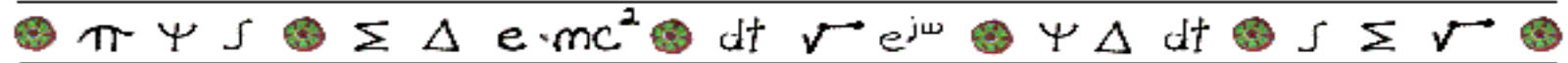
What is a Java Applet?

<http://javaboutique.internet.com/ABCMenuMan/>

<http://javaboutique.internet.com/TOV/>

<http://javaboutique.internet.com/billsClock/>

Morning Activity – Create Java Applets

A decorative horizontal line featuring a sequence of mathematical symbols and icons: a globe, pi, psi, integral, a globe, sigma, delta, E=mc^2, a globe, dt, a vector arrow, e^{j\omega}, a globe, psi, delta, dt, a globe, integral, sigma, a vector arrow, and a globe.

Demo of:

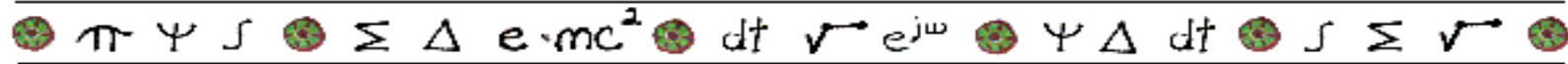
Ulead Animation Applet

Ulead Button Applet



Afternoon Activity: Programming in Java

Task 1: Printing to the screen

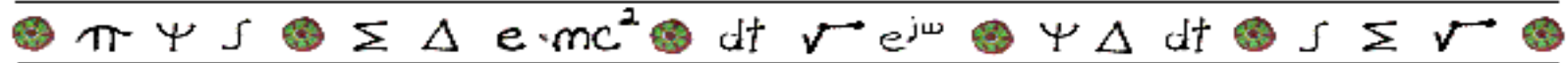


Open “sourcefile1.java” using Notepad
what words do you see?

Use the “system.out.println()” command to
print a statement

Demo of first task

Task 2: Variables and Computer Math



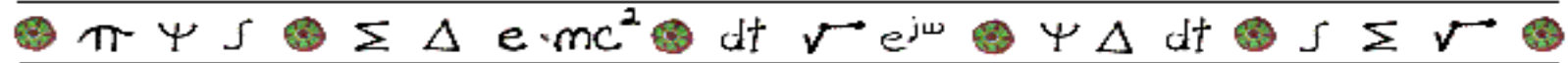
```
int result;
```

```
result = 3;
```

```
result = 4*3;
```

```
system.out.println(result);
```

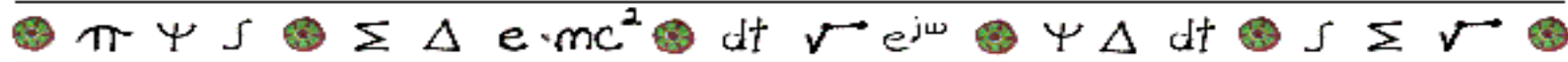
Task 3: Interactive Programming

A decorative horizontal line containing a sequence of mathematical symbols and icons: a globe, the Greek letter pi, the Greek letter psi, the Greek letter sigma, a globe, the Greek letter delta, the equation $e \cdot mc^2$, a globe, the differential dt , a vector arrow, the expression $e^{j\omega}$, a globe, the Greek letter psi, the Greek letter delta, the differential dt , a globe, the Greek letter sigma, a vector arrow, and a globe.

What is an example of an interactive program?

`argv[0]`, `argv[1]`...

Task 4: Printing to the screen



When does a computer program need to make a decision?

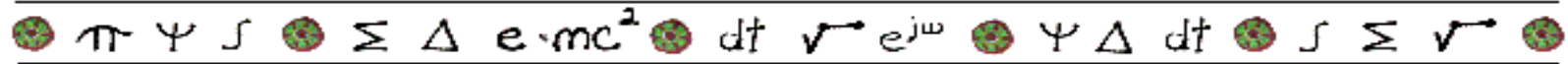
```
if (result == 2)
```

```
    system.out.println("result is 2!");
```

```
else
```

```
    system.out.println("result is not  
2!");
```

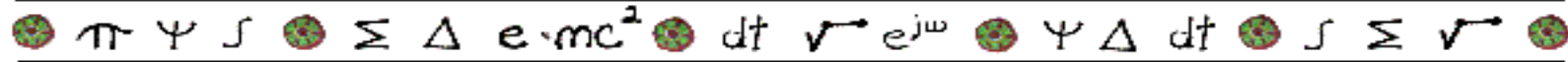
Task 5: For Loops



Why do you use for loops?

```
int i;  
for (i=0; i<20; i++)  
{  
    System.out.println("hello world.");  
}
```

Task 6: Print out a Graphical User Interface (GUI)









What is a GUI used for?

Size the window

Name the window

Task 7: Create Your Own Program

 π Ψ \int  Σ Δ $e \cdot mc^2$  dt $\sqrt{\quad}$ $e^{j\omega}$  Ψ Δ dt  \int Σ $\sqrt{\quad}$ 

Use the commands you've learned to create your own program!