Task: Errors and Data Types

www.hyperiondev.com

Introduction

Welcome to the Errors and Data Types Task
Overview:

You should now be quite comfortable with basic variable identification, declaration, and implementation. You should also be familiar with the process of writing basic code which adheres to the correct Java formatting to create a running program. This task is aimed at furthering your knowledge of types of variables to create more functional programs. You'll also be exposed to error handling and basic debugging in order to fix issues in your code, as well as the code of others - a skill which is extremely useful in a software development career!

-The Hyperion Team

Data types and all things variable

You've dealt with an *int,* but there are other data types in the integer class:

Integer data types in Java.

```
1 byte maxByte = 127;
2 short maxShort = 32767;
3 int maxInt = 2147483647;
4 long maxLong = 9223372036854775807;
```

When it comes to floating point numbers, we use *float* or *double:*

Correct floating point declaration and assignment.

```
1 double age = 10.5;
```

If we'd used a *float*, we would have to append the number with a f as a suffix, so 10.5 should be 10.5f as in:

The correct way to define floating point numbers of type `float`.

```
1 float age = 10.5f;
```

Casting - It's all about the change

Data type conversion (casting) can happen between two primitive types. There are two kinds of casting:

Implicit: casting operation is not required; the magnitude of the numeric value is always preserved. However, *precision* may be lost when converting from integer to floating point types:

Implicit casting (int is converted to long, casting is not needed).

```
1 int  i = 65;
2 long l = i;
```

Explicit: casting operation required; the magnitude of the numeric value may not be preserved:

Explicit casting (long is converted to int, casting is needed).

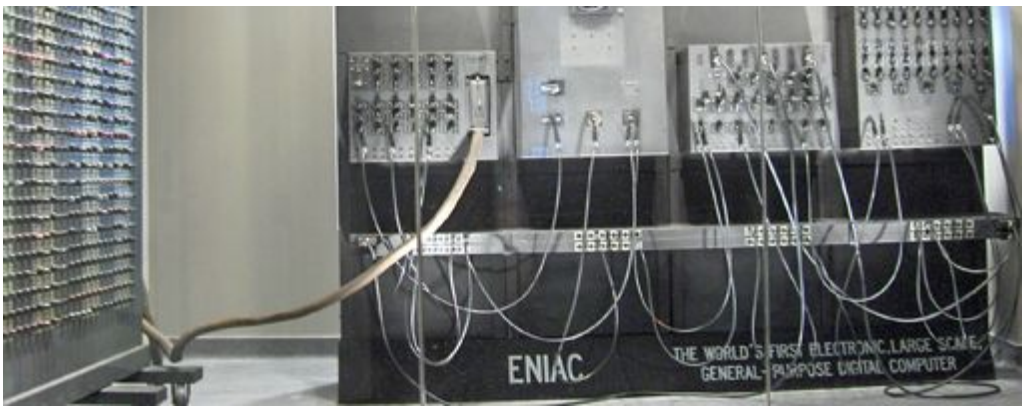
```
1 long l = 656666L;
2 int  i = (int) l;
```

Now that you've dealt with many different number definitions, you can see the mathematical impact of the number data types as such:

Category	Types	Size (bits)	Minimum Value	Maximum Value	Example
Integer	`byte`	8	-128	127	`byte b = 65;`
	`char`	16	0	$2^{16}-1$	`char c = 'A';` `char c = 65;`
	`short`	16	-2^{15}	$2^{15}-1$	`short s = 65;`
	`int`	32	-2^{31}	$2^{31}-1$	`int i = 65;`
	`long`	64	-2^{63}	$2^{63}-1$	`long l = 65L;`
Floating-point	`float`	32	2^{-149}	$(2-2^{-23})\cdot 2^{127}$	`float f = 65f;`
	`double`	64	2^{-1074}	$(2-2^{-52})\cdot 2^{1023}$	`double d = 65.55;`
Other	`boolean`	1	--	--	`boolean b = true;`
	`void`	--	--	--	--

Debuvic - whoops - Debugging

The term 'Debugging' comes from when bugs caused problems in computers - this was only possible when computers were as big as rooms! One of the first computers ever was known as ENIAC. This computer was located at the University of Pennsylvania. Riaz Moola, the Founder of Hyperion Development, recently studied at the University of Pennsylvania where ENIAC is still on display. Below is a picture of an ENIAC.



The principle of "try, try again" is a practice which software developers live by as this leads to the very best code implementation. This is done by the procedure of "testing and debugging", whereby you'll develop code, and run it on numerous occasions to hunt for errors or lack of performance in your program, and thereby editing or adding code to achieve this end goal!

Instructions

First read example.java, open it using jGRASP (Right click the file and select 'Open with jGRASP").

- In this folder there is a file called example.java
- Open this folder using the JGRASP program. You can either do this by right clicking on the example.java file, going to the 'Open with' menu, and selecting JGrasp.exe. Alternatively, you can run the JGRASP program on your computer, go to the top left corner of the program, select File->Open and navigate to example.java on your hard drive and double click on it to open it.
- Once example.java is open in JGRASP please read all of its content very carefully. This will help you understand the basic structure of a Java program.
- There is a compulsory task at the end of the example.java document. The instructions of what to do to complete the task can be found here. You must complete this exercise to continue onto the next task of this course.

Compulsory Task Part 1

Follow these steps:

- Open errors.java in your task folder.
- Attempt to compile the program.
- You'll notice errors when attempting to do this - fix the compilation errors and then correctly compile the program.
- Fix the runtime errors and then run the program.
- Save the file such that it runs correctly.
- Now run the program and notice the output - fix the logical error and indicate which of the three types of errors it was.
- Each time you fix an error, add a comment in the line you fixed it and indicate which of the three types of errors it was.

Compulsory Task Part 2

Follow these steps:

- Using jGRASP, create a new java file called 'part2'. (In jGRASP go to File -> New -> Java, then File -> Save as, and save it in this dropbox folder).
- Write code to define the outer class, followed by the main class. Remember that your outer class should be called part2.
- Inside your main class, write code to declare 5 *string* variables that each store words of your choice.
- Print these 5 strings using system.out.println statements
- Now define two *double* variables storing two numbers - 2 and 50.
- Now multiply the two variables, and store the result in a new *double* variable.
- Print out the resultant variable using system.out.println
- Compile and run your code.
- Use this task to test your knowledge of the syntax and your ability to correct errors you introduce.

Compulsory Task Part 3

Follow these steps:

- Using jGRASP, create a new java file called 'part3'. (In jGRASP go to File -> New -> Java, then File -> Save as, and save it in this dropbox folder)..
- Write code to define the appropriate outer and main classes.
- Write a statement to declare and assign a new *int* variable storing any number.
- Cast this variable into a new *string* variable called 'strInt'. Add a comment to this line of code, stating which type of casting you've carried out.
- Print out the 'strInt' variable.
- Next, declare a *string* variable called 'numStr', and store the "10" in it.
- Cast the *string* to a new *int* variable called 'numTen'. Again, add a comment to this line of code, stating which type of casting you've carried out.
- Finally, print out the 'numTen' variable.

Optional Task

Follow these steps:

- Create a new java file, called 'logic'.
- Within your logic.java file, write the java class and main class.
- Inside the main class, write a program that displays a logical error (be as creative as possible!).

Things to look out for

1. Make sure that you have installed and setup all programs correctly. You have setup **Dropbox** correctly if you are reading this, but **jGRASP or Java** may not be installed correctly.
2. If you are not using Windows, please ask your tutor for alternative instructions.

Still need help?

Just write your queries in your comments.txt file and your tutor will respond.

Task Statistics

Last update to task: 19/01/2016.
Authors: Riaz Moola and Jared Ping.
Main trainer: Umar Randeree.
Task Feedback link: Hyperion Development Feedback.