Task: Control Structures and Input

www.hyperiondev.com

Introduction

Welcome to the Control Structures and Input Task

Overview:

Now that you know how to debug programs, it's time to get started on some more complicated programming constructs - if you run into issues it's important that you remember how to resolve these compilation (runtime or logical) errors. Practice makes perfect, but even more so with programming! The tasks from now on are going to require you to write more programs to become familiar with the syntax, structure and style of Java programming - as well as using JGrasp to compile, run and debug programs.

Programming is more similar to cricket than it is to mathematics, physics or biology. Rather than memorising facts and laws, programming is more hands on - only through independent programming, solving problems by yourself and working outside of assigned exercises can you learn and understand how to be a good programmer. Only through using this method will programming become intuitive to you - like hitting a cricket ball. Fixing errors and typing statements with the correct syntax should become nothing more than a learnt reflex.

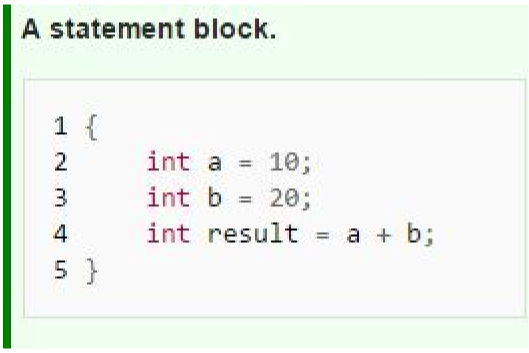
We encourage you to write your own programs and save them in your Dropbox folder. Call them anything you want - your trainer will look at them and can offer more specialised advice for you.

-The Hyperion Team

A note from the Hyperion Team...

Always wondering "IF" you should?

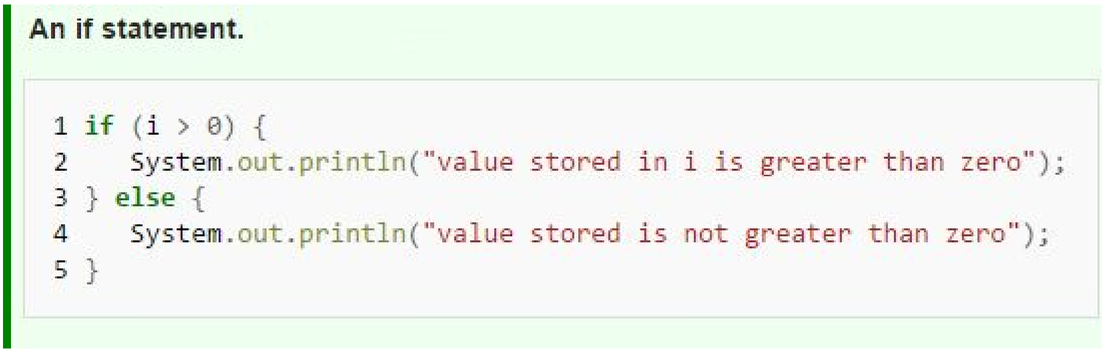
A bunch of statements can be placed in braces to be executed as a single block. Such a block of statement can be named or be provided a condition for execution. Below is how you'd place a series of statements in a block:



A statement block.

```
{
    int a = 10;
    int b = 20;
    int result = a + b;
}
```

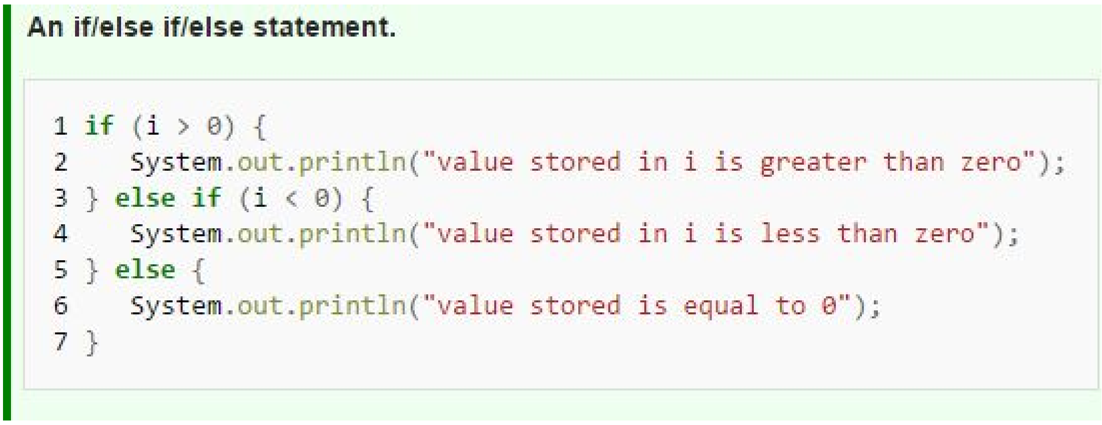
Conditional branching, also referred to as *if* statements, allows a program to perform a test and then take action based on the result of that test:



An if statement.

```
if (i > 0) {
   System.out.println("value stored in i is greater than zero");
} else {
   System.out.println("value stored is not greater than zero");
}
```

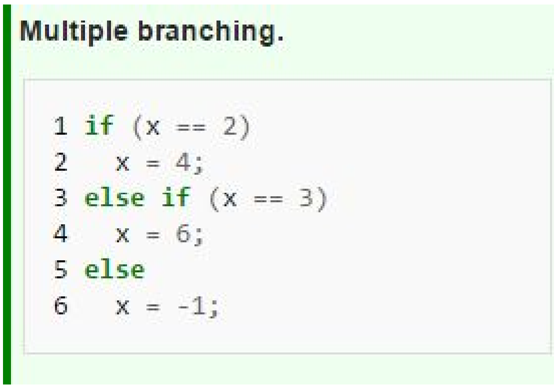
If statements can also be made more complex using the *else if* combination:



An if/else if/else statement.

```
if (i > 0) {
   System.out.println("value stored in i is greater than zero");
} else if (i < 0) {
   System.out.println("value stored in i is less than zero");
} else {
   System.out.println("value stored is equal to 0");
}
```

If...else statements also allow for the use of another statement, *else if*. This statement is used to provide another *if* statement to the conditional that can only be executed if the others are not true. For example:

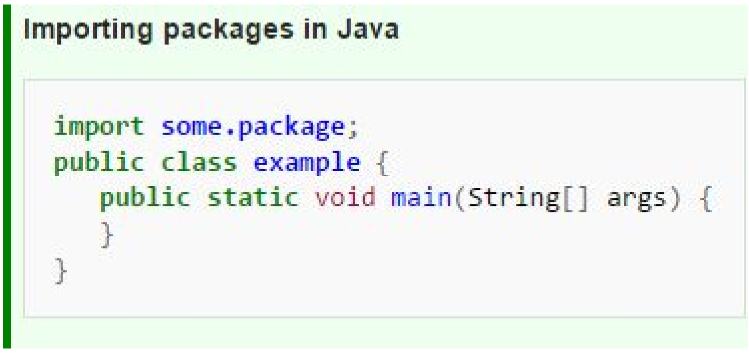


Multiple branching.

```
if (x == 2)
  x = 4;
else if (x == 3)
  x = 6;
else
  x = -1;
```

It's all about the packaging

The ability to use Java's built in functionality comes in the form of the import command, which retrieves commands, functions, or classes from prebuilt libraries:

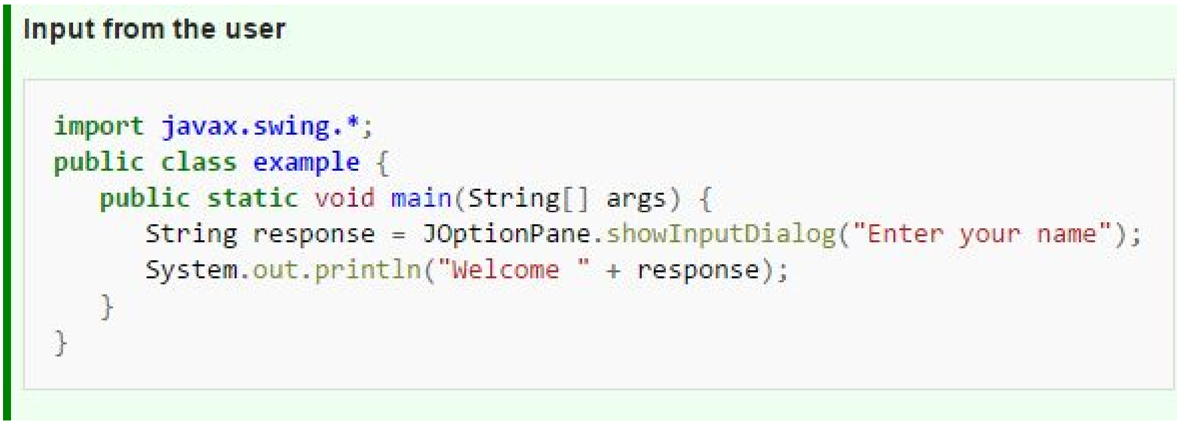


Importing packages in Java

```
import some.package;
public class example {
   public static void main(String[] args) {
   }
}
```

What does the user have to say?

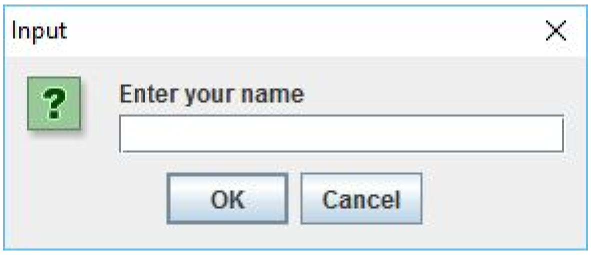
Now that you know how to import packages, it's essential to learn how to retrieve input from the user:



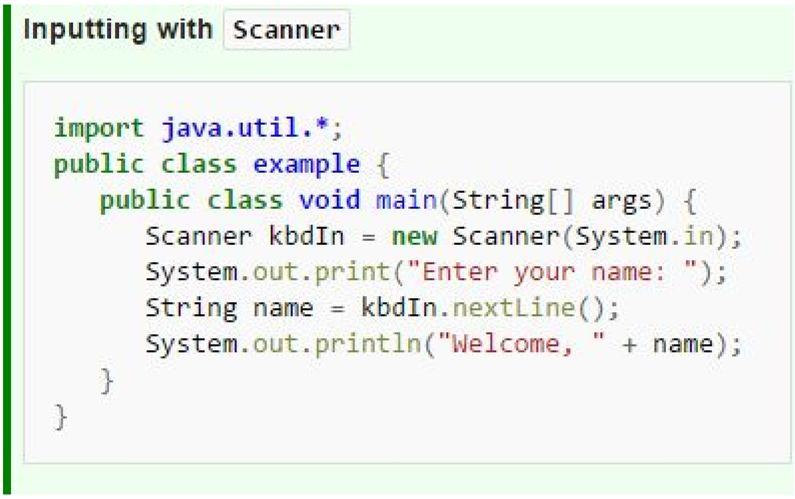
Input from the user

```
import javax.swing.*;
public class example {
   public static void main(String[] args) {
      String response = JOptionPane.showInputDialog("Enter your name");
      System.out.println("Welcome " + response);
   }
}
```

When you run the above code, you'll notice a dialogue box pop up as seen below:



It's worth noting that there's many ways of getting input from the user; it's also worth researching for your own benefit! One way is essentially a method which scans the user's keyboard for input and stops once the user hits the *enter* key - see if you can figure out how it works:



Inputting with `Scanner`

```
import java.util.*;
public class example {
   public class void main(String[] args) {
      Scanner kbdIn = new Scanner(System.in);
      System.out.print("Enter your name: ");
      String name = kbdIn.nextLine();
      System.out.println("Welcome, " + name);
   }
}
```

Don't worry if this doesn't make too much sense, it's quite above your expected knowledge at this point. As you become more knowledgeable and confident with your programming skills you'll find yourself venturing online through articles and content containing tricks and unique features of your programming language to play around with!

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 1

Follow these steps:

- Create a new file called firstIf.java.
- Write the outer class (remember to name it correctly) and the main class as shown before.
- Inside the main class, define a new variable called num and assign it any number.
- Below this, write an if else statement.
- The IF condition should be if the number is bigger than 10 - if true, print out "This number is bigger than 10"
- Else print out that the number is not bigger than 10.
- Compile, save and run your file.

Compulsory Task 2

Follow these steps:

- Create a new file called firstStringIf.java.
- Write the outer class (remember to name it correctly) and the main class as shown before.
- Inside the main class, define a new variable called name and assign it any string.
- Below this, write an if else statement.
- The IF condition should be if the string is equal to "Tim" - if true, print out "Welcome Tim".
- Else print out that the user is not recognised.
- Compile, save and run your file.

Compulsory Task 3

Follow these steps:

- Create a new file called firstDoubleStringIf.java.
- Write the outer class (remember to name it correctly) and the main class as shown before.
- Inside the main class, define two new string variable, nameOne and nameTwo. Assign both a name.
- Below this, write an if else statement.
- The IF condition should be if the strings are equal - if true, print out "Names are the same"
- Else print out that the names are not the same.
- Compile, save and run your file.

Compulsory Task 4

Follow these steps:

- Create a new file called ifElse.java.
- Write the outer class (remember to name it correctly) and the main class as shown before.
- Now write a program that defines a string stored in a variable called name.
- If the name is equal to Thabo, print out "Hello Thabo", otherwise if the name is equal to "Tony", print a greeting to Tony.
- Else print out user not recognised.
- Compile, save and run your file.
- 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 5

Follow these steps:

- Create a new file called operators.java.
- Write the outer class (remember to name it correctly) and the main class as shown before.
- Duplicate the code from line 324 - 346 in Example.java in your program. (Do not copy and paste code unless you are certain you understand the syntax of all the different operators).
- Change the value of numd to 120.
- Compile, save and run your code - ensure it works.
- Notice that when the value of numd is 120, multiple conditions of the structure are correct, but only one output is observed. Why is this? Look at the order of conditions and write a comment in your program explaining why this is. (Remember to us '//' in front of a line to make it a comment).
- Now change the value of numd to 30 and note the output - write a comment explaining the output.

Compulsory Task 6

Follow these steps:

- Create a new file called input.java.
- Write the outer class (remember to name it correctly) and the main class as shown before.
- Duplicate the import statement from Example.java and insert it in the correct place in your program.
- In your main class, write the JOptionPane statement and prompt the user with the message “What is your name?”.
- Store the user input in a string variable called input.
- Finally, print out the contents of the variable input.
- Compile, save and run your file multiple times with different inputs - note what happens and how it works.

Compulsory Task 7

Follow these steps:

- Create a new file called finalInput.java.
- Write the outer class (remember to name it correctly) and the main class as shown before.
- Copy and paste your code from Task 6.
- Write an if-elseif-else statement.
- Else if the user inputs the name “Ben”, print out “Greetings Ben”.
- Else print out “Greetings Stranger”.
- Run your program multiple times with different inputs - notice how you’ve made a dynamic program that gives different output depending on input?

Optional Task

Follow these steps:

- Create a new file called castInput.java.
- Write the outer class (remember to name it correctly) and the main class as shown before.
- Ask a user to input their age.
- Divide their age by 2 and output it.
- Compile, save and run your file.

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: 20/01/2016.
Authors: Riaz Moola and Jared Ping.
Main trainer: Umar Randeree.
Task Feedback link: Hyperion Development Feedback.