Hyperiondev

TASK

Introduction to Java Programming I - Java Basics

Visit our website

Introduction

Welcome to the Java Basics Task!

Java is a high-level programming language developed in 1995 by Sun Microsystems. This general purpose language is simple to learn, highly portable and is able to run on a variety of platforms. This task provides a brief overview of the fundamental concepts of Java such as variables, data types and control structures.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

A note from the

Hyperion Team

Some Java History:
Java was created primarily by a man named James Gosling, working at Sun Microsystems - now owned by the Oracle Corporation. The so called "Green Team" he lead had planned for the language to be the force to unite different consumer devices. It was designed to be lightweight so it could run using the devices' limited hardware while still being able to allow communication between devices. However once the Internet began to gain traction, the team realised their new language would be ideal for it, and shifted their focus.

Java's popularity is due to its small footprint on electronic devices, and its early boom into the technological world, means it has been given the chance to grow and flourish into the enormous creation that it is today.

So take a note from the Green Team, and keep in mind that even if you don't end up going where you had planned, it isn't always a bad thing!

Creating a Simple Java Program

The following Java program will display the message "Hello World!" on the console (the computer's display device).

```
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Hello World!");
        //prints out Hello World
    }
}
```

The first line in the above program, defines a class. Every Java program must have at least one class and every class has a name. The class defined above is given the name Welcome.

The second line of the program defines the main method. A method is a construct that contains statements. A class may contain many methods, however, the main method is the starting point where the program begins execution.

The main method above contains the System.out.println() statement on the third line which outputs the message "Hello World" to the console. Note that all statements in Java end with a semicolon (;), which is known as the statement terminator.

Line four of the program is a comment. Single line comments are preceded by double slashes (//), while multiline comments are enclosed between /* and */.

All of the program's components are grouped by using a pair of curly braces ({ }). Every class groups the data and methods of the class and every method groups the statements in the method.

Reading Input From the Console

Java uses System.out to refer to the standard output device, which is the computer's display monitor, and uses System.in to refer to the standard input device, which is the keyboard.

In the program above we use the System.out.println() method to print a message to the console, but what if we would like to read input from the console? In order to do this you need to create a Scanner object to read input from System.in. You do this by using the following Syntax:

```
Scanner input = new Scanner(System.in);
```

new Scanner(system.in) creates a Scanner object, while the syntax Scanner input declares a variable input of type Scanner. However, before creating a new Scanner object, we need to import the java.util package. In order to do this we use the following syntax:

```
import java.util.Scanner;
```

Variables

Variables are used to store values that you might wish to use later in the program. They are called variables because the values which the store can be changed.

A variable needs to be declared before it can be used. Declaring a variable simply informs the compiler how much of memory to allocate to the variable based on its data type. In order to declare a variable you need to enter the data type followed by the variable name.

An examples of a variable declarations can be found below:

```
int number;
double interestRate;
```

After the variable is declared it can be assigned a value by using the assignment operator (=).

```
number = 234;
interestRate = 4.56;
```

Named Constants

A named constant holds a permanent value that cannot be changed. In order to declare a constant you use the following syntax:

```
final double PI = 3.14159;
final int MAX_VALUE = 100;
```

Numeric Data Types

There are six numeric data types for integers and floating point numbers in Java. There are four integer types namely, byte, short, int and long, and two floating-point number types namely, float and double.

Name	Range	Storage Size (bits)
byte	-128 to 127	8
short	-2^{15} to $2^{15}-1$	16
int	-2^{31} to $2^{31}-1$	32
long	-2^{63} to $2^{63}-1$	64
float	Negative range: - 3.4028235E+38 to - 1.4E - 45 Positive range: 1.4E - 45 to 3.4028235E + 38	32
double	Negative range: - 1.7976931348623157E + 308 to - 4.9E - 324 Positive range: 4.9E - 324 to 1.7976931348623157E + 308	64

Casting

You are able to convert one data type to another using casting. There are two types of casting in Java, implicit and explicit.

Implicit casting occurs automatically when you assign a value to a variable whose type supports a larger range of values. For example, you can assign an int value to a float. This is known as widening a type and you do not need any code in order to do this.

An example of implicit casting is as follows:

```
int number1 = 5;
float number2 = number1;
```

Casting a type with a large range to a type with a smaller range is known as narrowing a type, and you must use explicit casting in order to do this. In order to explicitly cast a type you need to specify the target type in parentheses, followed by either the variable's name or value you wish to cast.

An example of explicit casting is as follows:

```
int number1 = (int)23.34;

long longNum = 656666L;
int intNum = (int) longNum;
```

The Character and String Data Types

In Java you can represent a single character using the character data type, char. All character values must be enclosed between single quotation marks, for example:

```
char letter = 'N';
char number = '5';
```

You can also represent a string of characters using the data type String. Like many other programming languages all strings must be enclosed between double quotation marks, for example:

```
String newString = "Java is fun!";
```

The Boolean Data Type

The boolean data type can only hold one of two possible values, true or false. Boolean values are the result of comparison operations. For example the following statement will display true:

```
int number = 10;
System.out.println(number < 100);
```

if-else Statements

As you know, selection statements, such as if statements, play a pivotal role in programming. Selection statements allow the program to decide which statements to execute based on a condition.

An if statement will execute a block of code only if the condition is true. The syntax for an if statement in Java is as follows:

```
if(boolean expression) {
    statement(s);
}
```

But what happens if the condition is false? If you simply had an if statement nothing would happen. If you want to take alternative actions when the condition is false you can add an else statement to your if statement to create an if-else statement. If the condition turns out to be false the statements indicated by the else statement will be executed. The syntax for an if-else statement is as follows:

```
if(boolean expression) {
    statement(s);  // executed if boolean expression evaluates to true
}
else {
    statement(s);  // executed if boolean expression evaluates to false
}
```

An if or if-else statement can also placed within another if or if-else statement to form a nested statement.

For example the following code tests what symbol a student should be awarded based on their class mark. If the student achieves a mark of 80 or greater they receive an A, else if they achieve a mark of 70 or greater they receive a B, and so on.

```
if (mark >= 80)
    mark = 'A';
else if (mark >= 70)
    mark = 'B';
else if (mark >= 60)
    mark = 'C';
else if (mark >= 50)
    mark = 'D';
else
    mark = 'F';
```

The while Loop

Loops are used to control how many times a statement or sequence of statements are executed. The while loop is a simple loop that executes statements repeatedly while a condition is true.

The syntax for the while loop is:

```
while (continuation condition) {
    statement(s);
}
```

To avoid an infinite loop (loop that never terminates) you must make sure that the the continuation condition will eventually become false and the loop stops running.

The for Loop

The for loop provides a concise, simple way to iterate over a range of values.

The syntax for the for loop is:

```
for (initialisation; termination; increment) {
```

```
    statement(s);
}
```

The initialisation expression is executed once when the loop begins. It is normally used to initialise a control variable. Next the termination expression tests whether the control variable has reached its termination value, if it has not the statements in the body of the loop are executed. The increment expression then increments or decrements the control variable and the termination value is tested once again. The process is repeated until the variable has finally reached its termination value.

Compulsory Task 1

Follow these steps:

- Create a new file called rockPaperScissors.java
- Write a program that allows the user to play the rock, paper, scissors game.
- The program should randomly generate a number 0, 1, 2, which represents scissors, rock and paper respectively.
- The program should then prompt the user to enter a number 0, 1, 2.
- Once the user has entered their number the program should inform them whether they win, lose or draw.
- The rules of the game are as follows:
    - Scissors beats paper
    - Rock beats scissors
    - Paper beats rock

Compulsory Task 2

Follow these steps:

- Create a new file called averageNumber.java

- Write a program that determines how many positive and negative integers have been entered and calculates the total and average of all the entered numbers.
- Firstly ask the user to enter any number of integers. The user should enter 0 to indicate the end of their input.
- The program should then determine the number of positive and negative integers entered by the user, and print out the result.
- The total of all integers entered as well as the average should then be calculated and displayed.

Compulsory Task 3

Follow these steps:

- Create a new file called factorial.java
- Write a program that determines the factorial of a number entered by a user.
- Prompt the user to enter a positive integer.
- Then calculate the factorial of the given number. For any positive integer n, it's factorial is given by:
    - factorial = 1*2*3...*n
- Finally, print out the factorial.

Rate us

Share your thoughts

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.