**Homework 9 Ankur Jain Submit to:** [**Homework@PeopleNTech.Com**](mailto:Homework@PeopleNTech.Com)

**Subject of Email: “AnkurJain\_StudentID \_First Name, Last Name – Homework 9”**

**TOPICS: Java/Selenium/JIRA/Git/GitGub/HTML/POSTMAN/SOAPUI/TestNG**

**Deadline : March 20th**

Short Paper :

Research on following topic and build 4-5 pages short paper:

1. Loops and Data types.
2. History of Java Releases
3. The Java Program Life Cycle
4. Advantages of Java programming language
5. Java Primitive data type
6. Java Development Environment Setup

**1-What are Loops and Date Types?**

* **Without Loops:**

public static void main (String[] args) {

System.out.println(“software testing help blog”);

System.out.println(“software testing help blog”);

System.out.println(“software testing help blog”);

System.out.println(“software testing help blog”);

System.out.println(“software testing help blog”);

}

* **With Loops:**

|  |
| --- |
| public static void main (String [] args) {    for (int i=0; i<5; i++) {  System.out.println(“software testing help blog”);  }  } |

**For loop is useful for iterating values.**

**For Loop:**

|  |
| --- |
| for (initialization; termination; increment) {  statement(s)  } |

**While Loop:**

While Loop checks the Boolean condition first and if that condition is true, the statements will be executed repeatedly until the condition fails.

|  |
| --- |
| While (condition) {  Statement to be executed  } |

**Do While Loop:**

In the do-while loop, the statement will be executed first and then the condition will be checked. If the condition is true, the statement will be executed again. Otherwise, execution will come out the loop.

|  |
| --- |
| do {  Statements to be executed  }  while(condition) {  Statements to be executed  } |

**2-What is the history of JAVA releases?**

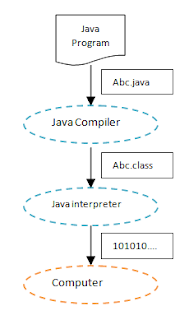
|  |  |  |
| --- | --- | --- |
| **Version Name** | **Code Name** | **Release Date** |
| **JDK 1.0** | Oak | January 1996 |
| **JDK 1.1** | (none) | February 1997 |
| **J2SE 1.2** | Playground | December 1998 |
| **J2SE 1.3** | Kestrel | May 2000 |
| **J2SE 1.4** | Merlin | February 2002 |
| **J2SE 5.0** | Tiger | September 2004 |
| **JDK 6** | Mustang | December 2006 |
| **JDK 7** | Dolphin | July 2011 |
| **JDK 8** |  | March 2014 |
| **JDK 9** |  | September, 21st 2017 |
| **JDK 10** |  | March, 20th 2018 |
| **JDK 11** |  | September, 25th 2018 |
| **JDK 12** |  | March, 19th 2019 |
| **JDK 13** |  | September, 17th 2019 |
| **JDK 14** |  | March, 17th 2020 |

Major versions were released after every 2 years however the Java SE 7 took 5 years to be available after its predecessor Java SE 6, and 3 years for Java SE 8 to be available to public afterward.

Since Java SE 10,  new versions will be released very six months.

**3-What is the JAVA Program Life Cycle?**

1. A Java program is written using either a Text Editor like Textpad or an IDE like Eclipse and is saved as a .java file. (Program.java)
2. The .java file is then compiled using Java compiler and a .class file is obtained from it. (Program.class)
3. The .class file is now portable and can be used to run this Java program in any platform.
4. Class file (Program.class) is interpreted by the JVM installed on a particular platform. JVM is part of the JRE software.

[](http://2.bp.blogspot.com/-f8cY6lX0r-g/TsJ94YebJGI/AAAAAAAAARo/9XEfqxGfrcE/s1600/Java1.png)

**4-What are the advantages of JAVA programming language?**

**Advantages of Java** are: Simple: **Java** was designed to be easy to use, write, compile, debug, and learn than other **programming languages**. Object-Oriented: Allows you to create modular programs and reusable **code**. Platform-Independent: Ability to move easily from one computer system to another.

**5-What is a JAVA primitive type?**

# Primitive Data Types

The Java programming language is statically-typed, which means that all variables must first be declared before they can be used. This involves stating the variable's type and name, as you've already seen:

int gear = 1;

Doing so tells your program that a field named "gear" exists, holds numerical data, and has an initial value of "1". A variable's data type determines the values it may contain, plus the operations that may be performed on it. In addition to int, the Java programming language supports seven other *primitive data types*. A primitive type is predefined by the language and is named by a reserved keyword. Primitive values do not share state with other primitive values. The eight primitive data types supported by the Java programming language are:

* **byte**: The byte data type is an 8-bit signed two's complement integer. It has a minimum value of -128 and a maximum value of 127 (inclusive). The byte data type can be useful for saving memory in large [arrays](https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html), where the memory savings actually matters. They can also be used in place of int where their limits help to clarify your code; the fact that a variable's range is limited can serve as a form of documentation.
* **short**: The short data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive). As with byte, the same guidelines apply: you can use a short to save memory in large arrays, in situations where the memory savings actually matters.
* **int**: By default, the int data type is a 32-bit signed two's complement integer, which has a minimum value of -231 and a maximum value of 231-1. In Java SE 8 and later, you can use the int data type to represent an unsigned 32-bit integer, which has a minimum value of 0 and a maximum value of 232-1. Use the Integer class to use int data type as an unsigned integer. See the section The Number Classes for more information. Static methods like compareUnsigned, divideUnsigned etc have been added to the [Integer](https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html) class to support the arithmetic operations for unsigned integers.
* **long**: The long data type is a 64-bit two's complement integer. The signed long has a minimum value of -263 and a maximum value of 263-1. In Java SE 8 and later, you can use the long data type to represent an unsigned 64-bit long, which has a minimum value of 0 and a maximum value of 264-1. Use this data type when you need a range of values wider than those provided by int. The [Long](https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html) class also contains methods like compareUnsigned, divideUnsigned etc to support arithmetic operations for unsigned long.
* **float**: The float data type is a single-precision 32-bit IEEE 754 floating point. Its range of values is beyond the scope of this discussion, but is specified in the [Floating-Point Types, Formats, and Values](https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3) section of the Java Language Specification. As with the recommendations for byte and short, use a float (instead of double) if you need to save memory in large arrays of floating point numbers. This data type should never be used for precise values, such as currency. For that, you will need to use the [java.math.BigDecimal](https://docs.oracle.com/javase/8/docs/api/java/math/BigDecimal.html) class instead. [Numbers and Strings](https://docs.oracle.com/javase/tutorial/java/data/index.html) covers BigDecimal and other useful classes provided by the Java platform.
* **double**: The double data type is a double-precision 64-bit IEEE 754 floating point. Its range of values is beyond the scope of this discussion, but is specified in the [Floating-Point Types, Formats, and Values](https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3) section of the Java Language Specification. For decimal values, this data type is generally the default choice. As mentioned above, this data type should never be used for precise values, such as currency.
* **boolean**: The boolean data type has only two possible values: true and false. Use this data type for simple flags that track true/false conditions. This data type represents one bit of information, but its "size" isn't something that's precisely defined.
* **char**: The char data type is a single 16-bit Unicode character. It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65,535 inclusive).