# Program No.1: Install and Setup java environment .Install java editor (Eclipse for Enterprise Java) and configure workspace. Execution of first java program. Java code execution process.

**Java environment setup:**

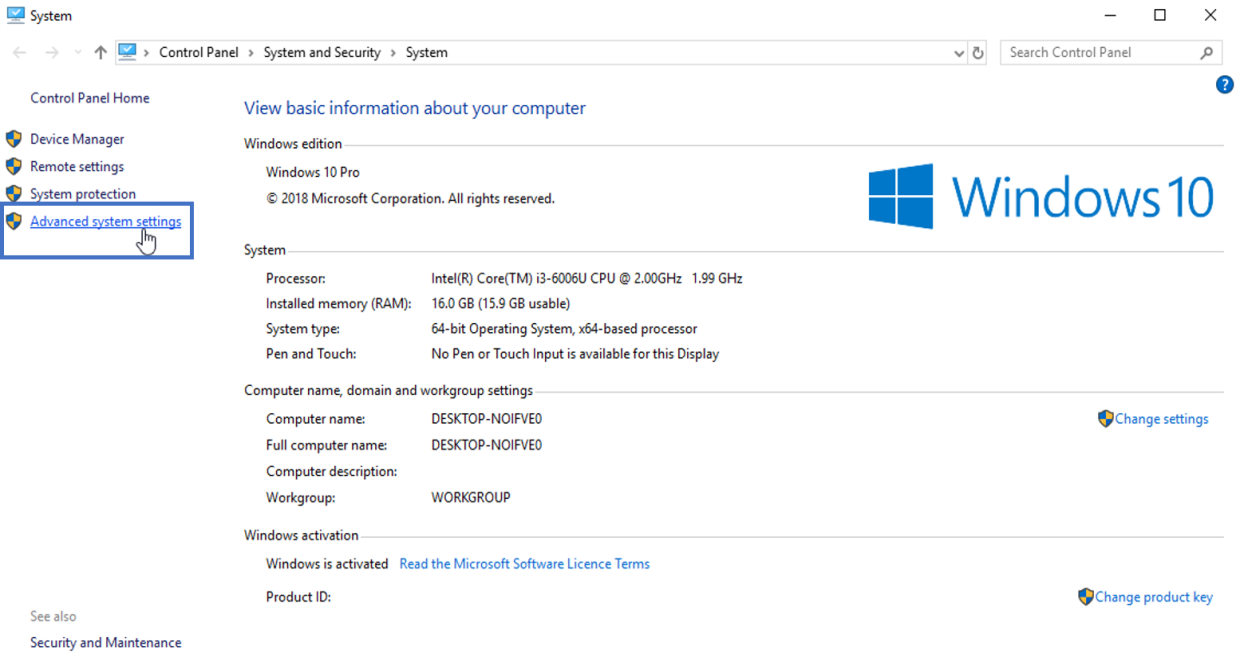
In today’s world, Java is one of the most popular programming language which is being used by most skilled professionals. However, using it on the command line is not feasible sometimes. Therefore, to overcome this, we can use it on **Eclipse IDE.** Let’s see how to setup Java environment on Eclipse IDE.

Step 1: Install Java

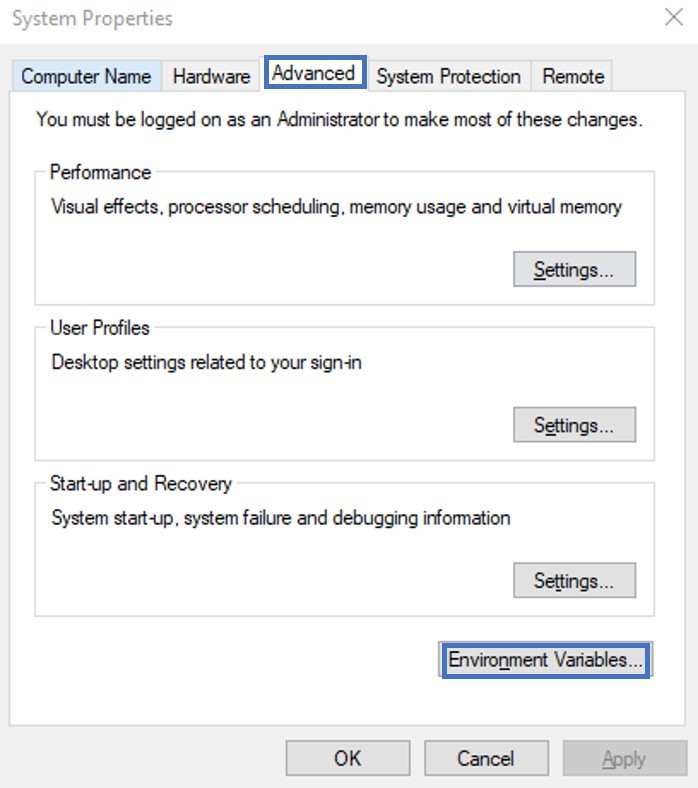
Step 2: Setup Eclipse IDE on Windows Step 3: Hello World Program

**Install Java:**

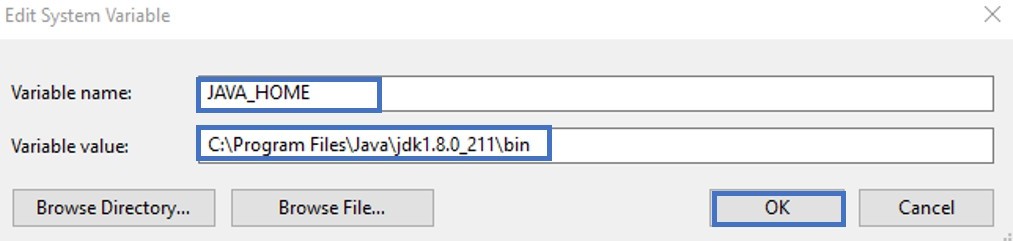
* Follow the below steps to complete your Java installation.
* Go to the Java Downloads Page and click on the option of **Download.**
* Now, once the file is downloaded, run the installer and keep clicking on **Next**, till you finally get a dialog box, which say, you have finished installing.
* Once the installation is over follow the below instructions to set the path of the file.
* Now right click on ThisPC/ My Computer Icon-> Go to its properties and its **Advanced System Settings**. Refer below.



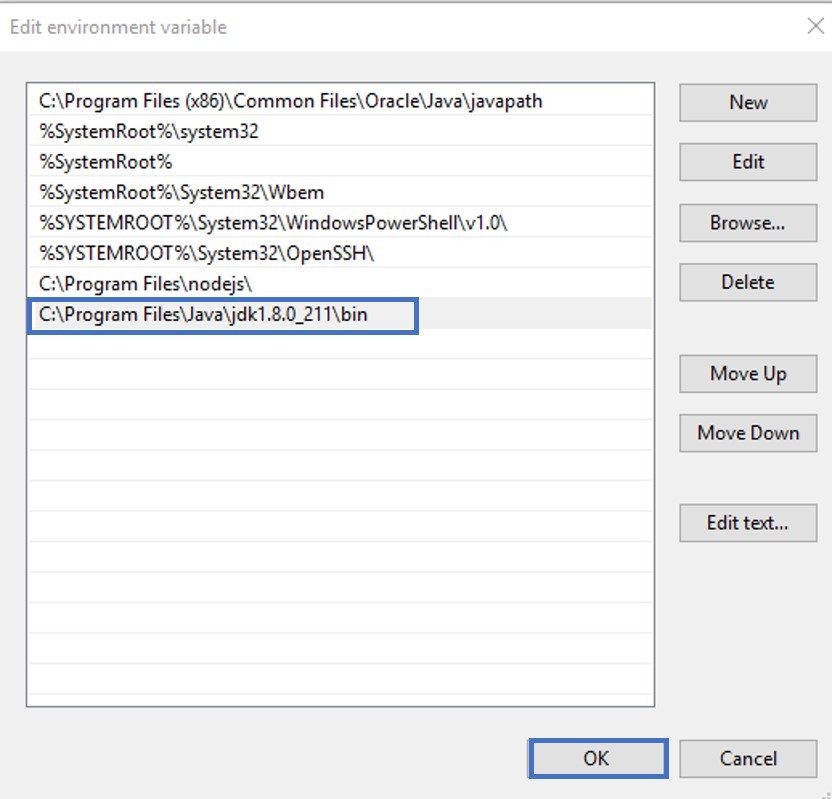
* Now, click on ‘**Environment Variables**’ under ‘**Advanced**’ tab as shown below:



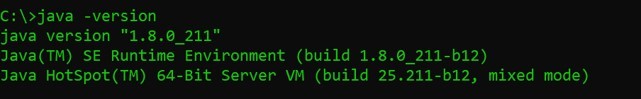
* Next, under **System Variables** choose **New.** Enter the variable name as ‘**JAVA\_HOME**’ and the full path to Java installation directory as per your system as shown below:



* Next thing that you have to do is to configure your environment variables. Let’s see how to do that. Here, you have to edit the path of the system variable as shown below. Then click OK.



* Now to cross-check the installation, just run following command in cmd – ***java -version***. It should display the installed version of Java in your system.

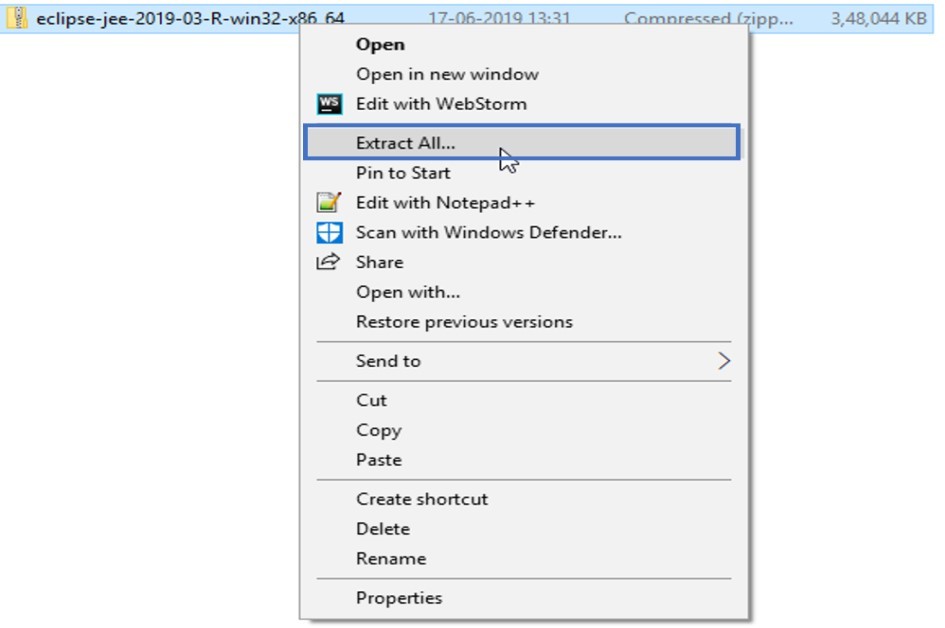


**Install Eclipse:**

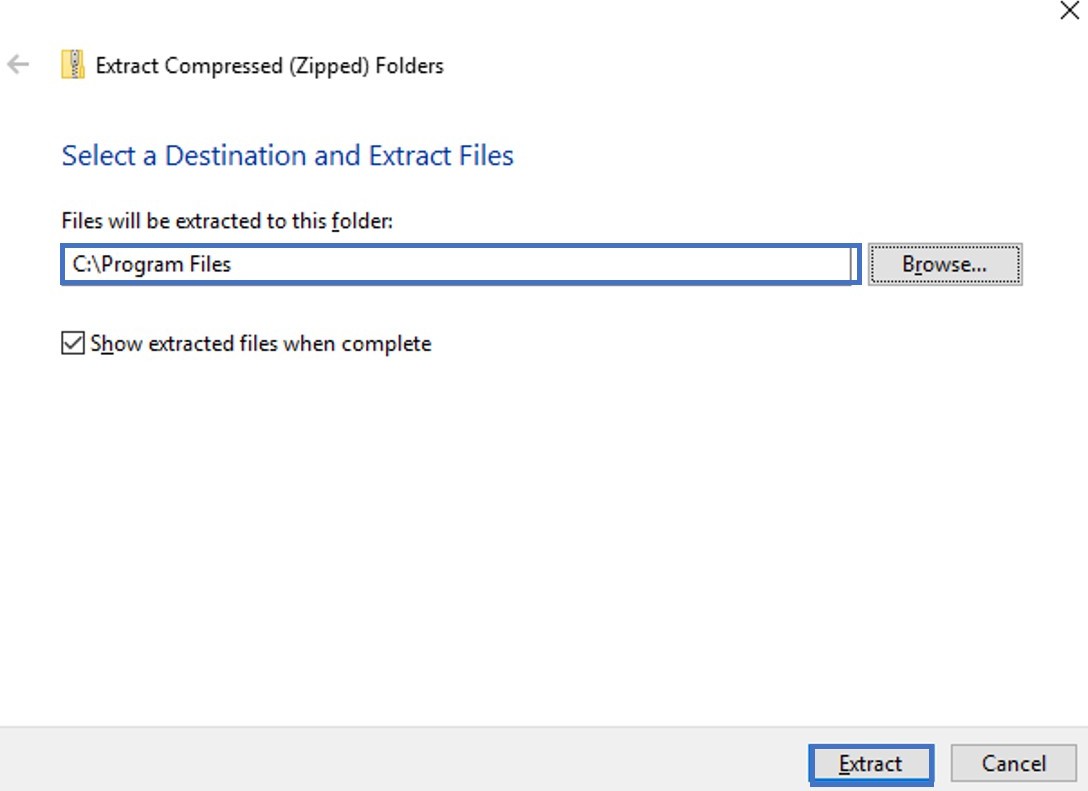
Follow the below steps to configure Eclipse on your system:

**Step 1:** Navigate to the following URL – https://[www.eclipse.org/downloads/packages/](http://www.eclipse.org/downloads/packages/) and select the download link depending on your system architecture – (Windows, Mac OS or Linux) and download it.

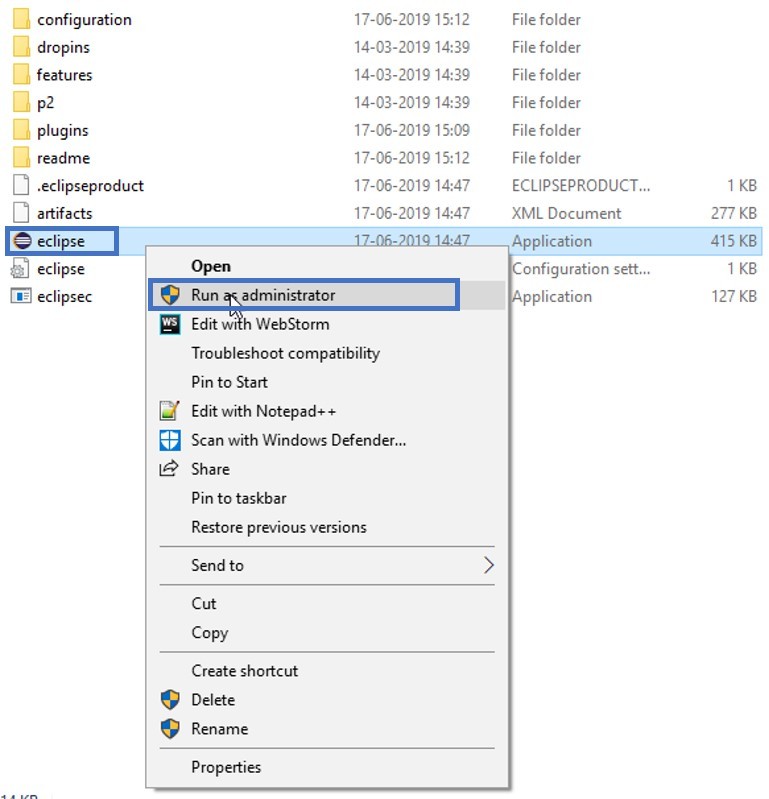
**Step 2:** Once the download is over, extract the zipped file by right-clicking on the folder and choose **Extract All**. Refer below.



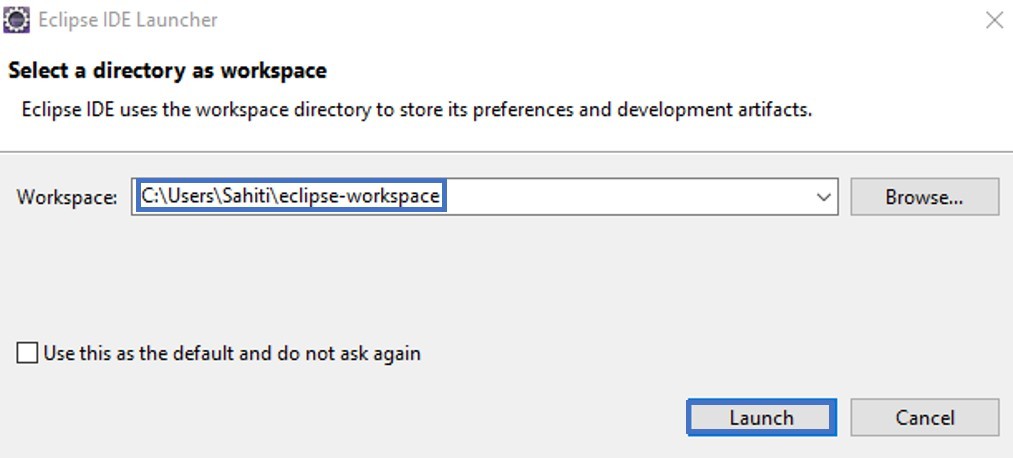
**Step 3:** You will be then redirected to a dialog box, where you have to choose the directory in which you wish to extract the files. Then click on **Extract**. Refer below.

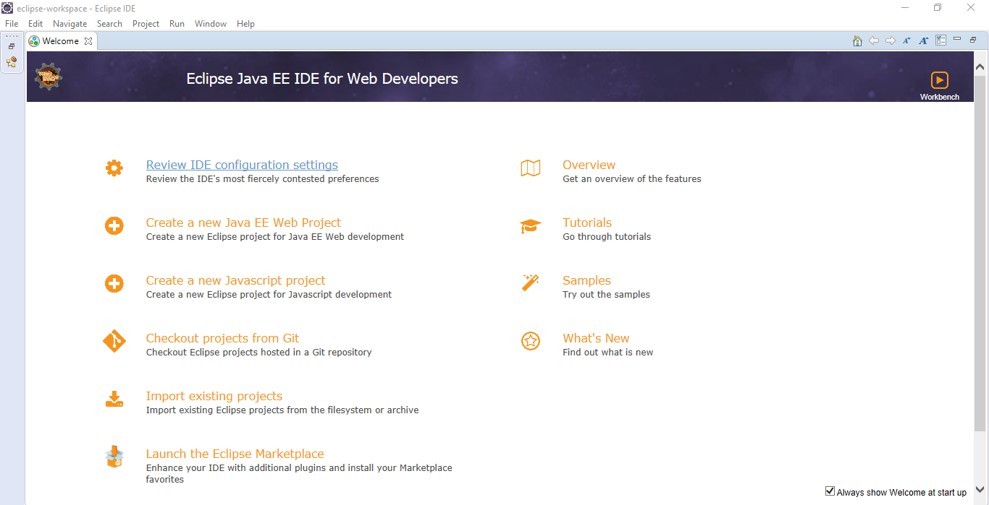


**Step 4:** After extracting files, open the folder and launch **eclipse.exe.**



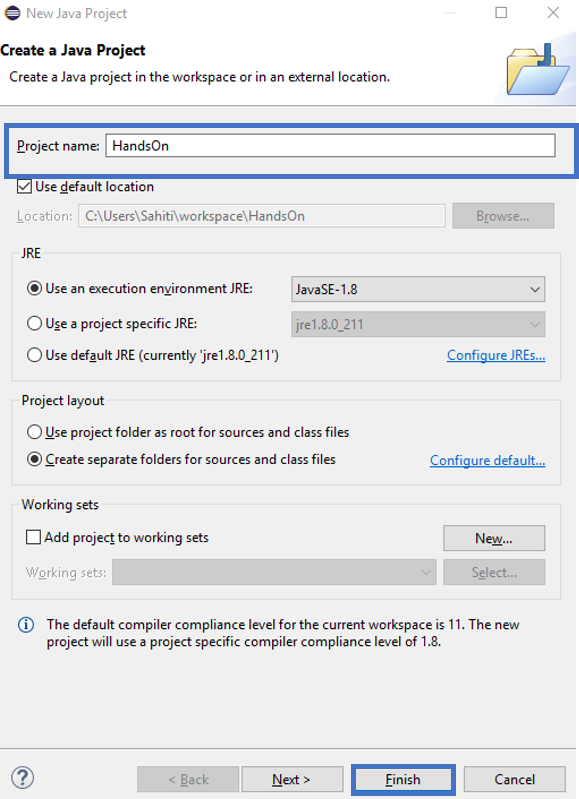
**Step 5:** Then, you have to choose the Launch directory for Eclipse and then click on Launch. Refer below.



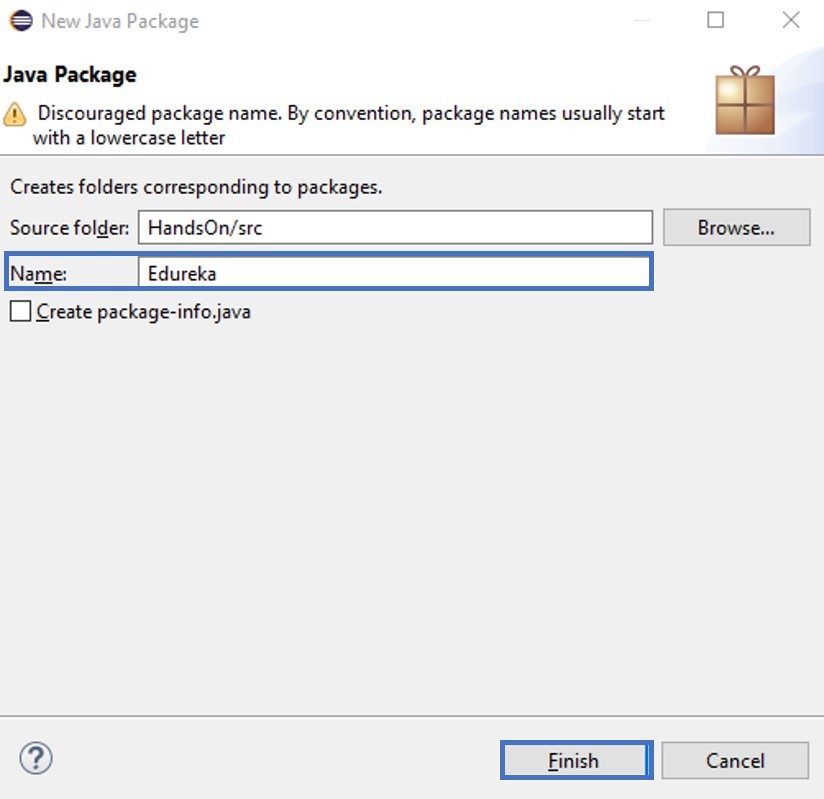
Step 6: Once Eclipse launches, you will see the below window:

**Executing first Java Program : Hello World Program:**

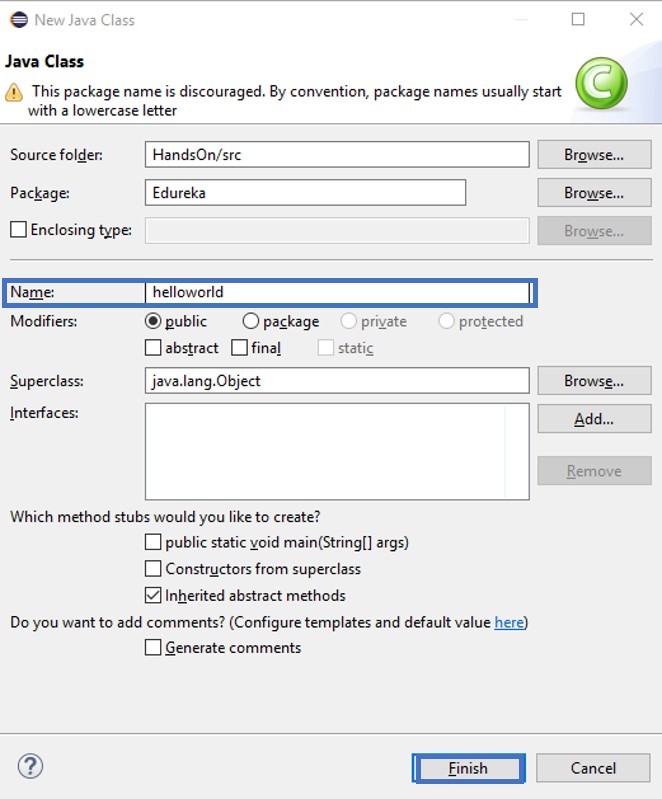
**Step 1:** Launch **Eclipse IDE** and go to **File ->New -> Java Project Step 2:** Mention the **project name** and click on **Finish.**



**Step 3:** Now, go to the **Project**, Right-Click on the Project and choose **Package**. In the dialog box, which opens up, mention the **Package name** as below and click on **Finish.**

****

**Step 4:** Now, right click on the **Package**, go to **New** and choose **Class**. Mention the **class name** and click on **Finish**. Refer below.



**Step 5:** Now, mention the following code in the workspace.

public class helloworld

{

public static void main(String args[])

{

System.out.println("Hello World");

}

}

**Step 6:** Now, execute your file, by right-clicking on the **helloworld.java file** and choose **Run As -> Java Application.**

# Program No.2: Code, execute and debug programs that uses different types of variables and datatypes; Identify and resolve issues in the given code snippet.

**class** Datatypes

{

**public static void** main(String[] args)

{

**byte** myByte1,myByte2; myByte1 = 120;

myByte2 = -48; System.*out*.println("Byte1: " +myByte1); System.*out*.println("Byte2: " +myByte2);

myByte1++; // Looping back within the range System.*out*.println("Incremented Value of Byte1:" +myByte1);

**short** myShort = 6000; System.*out*.println("\nShort:" +myShort);

**int** myInteger1, myInteger2, result; myInteger1 = -7000;

myInteger2 = 9000;

result = myInteger1 + myInteger2; System.*out*.println("\nInteger1:"+myInteger1); System.*out*.println("Integer2:"+myInteger2); System.*out*.println("Integer1 + Integer2: " +result);

**long** myLong1, myLong2, result1; myLong1 = 100000000L;

myLong2 = 200L;

result1 = myLong1 \* myLong2; System.*out*.println("\nLong1: " +myLong1); System.*out*.println("Long2: " +myLong2); System.*out*.println("Long1 \* Long2: " +result1);

**float** myFloat1,myFloat2,result2; myFloat1=1000.666f; myFloat2=110.77f; result2=myFloat1-myFloat2;

System.*out*.println("\nFloat1: "+myFloat1); System.*out*.println("Float2: "+myFloat2); System.*out*.println("Float1-Float2: "+result2);

**double** myDouble1, myDouble2, result3; myDouble1 = 48976.8987;

myDouble2 = 29513.7812d;

result3 = myDouble1 + myDouble2; System.*out*.println("\nDouble1: " +myDouble1); System.*out*.println("Double2: " +myDouble2); System.*out*.println("Double1 + Double2: " +result3);

**boolean** myBool = **true**; **if**(myBool == **true**)

System.*out*.println("\nI am using a Boolean data type"); System.*out*.println(myBool);

**char** myChar1 = 'A';

**char** myChar2 = 66; System.*out*.println("\nmyChar1: " +myChar1); System.*out*.println("myChar2: " +myChar2); myChar2++; // valid increment operation

System.*out*.println("The Incremented value of myChar2: " +myChar2);

String string1 = "\nGPT MUDHOL"; // declaring string using string literal System.*out*.println(string1);

}

}

**Output:**

Byte1: 120

Byte2: -48

Incremented Value of Byte1:121 Short: 6000

Integer1:-7000 Integer2:9000

Integer1 + Integer2: 2000

Long1: 100000000

Long2: 200

Long1 \* Long2: 20000000000

Float1: 1000.666

Float2: 110.77

Float1-Float2: 889.896

Double1: 48976.8987

Double2: 29513.7812

Double1 + Double2: 78490.6799

I am using a Boolean data type true

myChar1: A myChar2: B

The Incremented value of myChar2: C

GPT MUDHOL

# Program No.3: Code, execute and debug programs that uses different types of constructors. Identify and resolve issues in the given code snippet.

**class** Student

{

String name;

**int** regno;

Student() //Constructor

{

name="Raju"; regno=1234;

}

Student(String n, **int** r) // parameterized constructor

{

name=n; regno=r;

}

Student(Student s) // copy constructor

{

name=s.name; regno=s.regno;

}

**void** display()

{

System.*out*.println(name + "\t" +regno);

}

}

**class** StudentDemo

{

**public static void** main(String args[])

{

}

}

**Output:**

Student s1=**new** Student();

Student s2=**new** Student("Ravi",1489); Student s3=**new** Student(s1); s1.display();

s2.display();

s3.display();

Raju 1234

Ravi 1489

Raju 1234

# Program No.4: Code, execute and debug program to perform autoboxing and unboxing. Identify and resolve issues in the given code snippet.

**class** Conversion

{

**public static void** main(String args[])

{

**byte** b=10; **short** s=20; **int** i=30; **long** l=40;

**float** f=50.0F; **double** d=60.0D; **char** c='a'; **boolean** b2=**true**;

//Autoboxing: Converting primitives into objects Byte byteobj=b;

Short shortobj=s; Integer intobj=i; Long longobj=l; Float floatobj=f; Double doubleobj=d; Character charobj=c; Boolean boolobj=b2;

//Printing objects

System.*out*.println("---Printing object values---"); System.*out*.println("Byte object: "+byteobj); System.*out*.println("Short object: "+shortobj); System.*out*.println("Integer object: "+intobj); System.*out*.println("Long object: "+longobj); System.*out*.println("Float object: "+floatobj); System.*out*.println("Double object: "+doubleobj); System.*out*.println("Character object: "+charobj); System.*out*.println("Boolean object: "+boolobj);

//Unboxing: Converting Objects to Primitives

**byte** bytevalue=byteobj; **short** shortvalue=shortobj; **int** intvalue=intobj;

**long** longvalue=longobj;

**float** floatvalue=floatobj; **double** doublevalue=doubleobj; **char** charvalue=charobj; **boolean** boolvalue=boolobj;

//Printing primitives

System.*out*.println("---Printing primitive values---"); System.*out*.println("byte value: "+bytevalue); System.*out*.println("short value: "+shortvalue);

System.*out*.println("int value: "+intvalue); System.*out*.println("long value: "+longvalue); System.*out*.println("float value: "+floatvalue); System.*out*.println("double value: "+doublevalue); System.*out*.println("char value: "+charvalue); System.*out*.println("boolean value: "+boolvalue);

}

}

# Output:

---Printing object values--- Byte object: 10

Short object: 20

Integer object: 30

Long object: 40

Float object: 50.0

Double object: 60.0 Character object: a Boolean object: true

---Printing primitive values--- byte value: 10

short value: 20

int value: 30

long value: 40

float value: 50.0

double value: 60.0 char value: a boolean value: true

# Program No.5: Code, execute and debug program to perform evaluation of expression. Identify and resolve issues in the given code snippet.

import java.util.Scanner; class ExpressionEvaluation

{

public static void main(String[] args)

{

Scanner input = new Scanner(System.*in*); System.*out*.print("Enter Equation you want to evaluate : "); String string = input.nextLine();

String a = string.replaceAll(" ","");

if (a.length() < 3)

{

System.*out*.println( "Please Enter Minimum One Opearator and Two Opearands"); System.*exit*(0);

}

int result = 0; int i = 0;

while(a.charAt(i)!='+' && a.charAt(i)!='-' && a.charAt(i)!='\*' && a.charAt(i)!='/')

{

i++;

}

switch (a.charAt(i))

{

case '+' :

result = Integer.*parseInt*(a.substring(0,i))+Integer.*parseInt*(a.substring(i+1,a.length())); break;

case '-' :

result = Integer.*parseInt*(a.substring(0,i))-Integer.*parseInt*(a.substring(i+1,a.length())); break;

case '\*' :

result = Integer.*parseInt*(a.substring(0,i))\*Integer.*parseInt*(a.substring(i+1,a.length())); break;

case '/' :

result = Integer.*parseInt*(a.substring(0,i))/Integer.*parseInt*(a.substring(i+1,a.length())); break;

}

System.*out*.println(a.substring(0,i) + ' ' + a.charAt(i) + ' ' + a.substring(i+1,a.length())+ " = " + result);

}

}

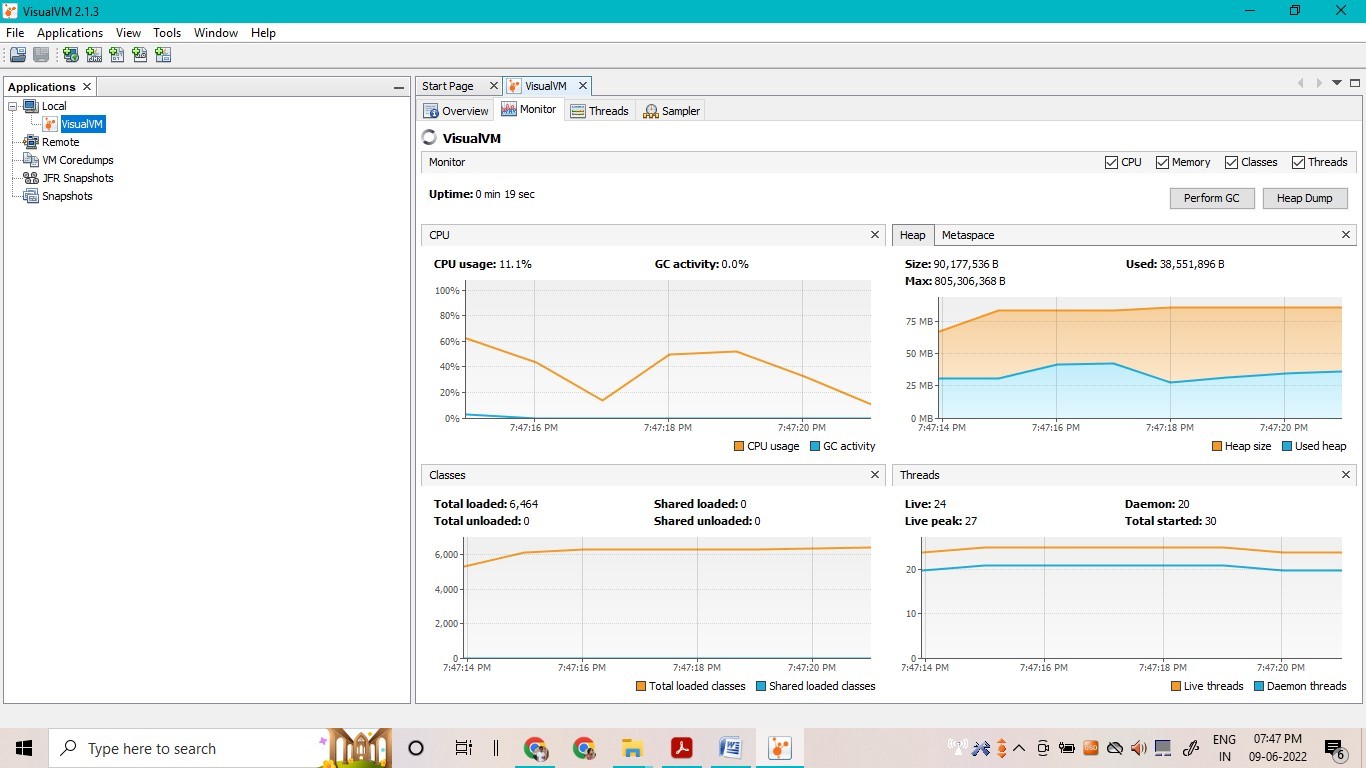
**Output:**

Enter Equation: 10+20

10 + 20 = 30

# Program No.6: Install memory monitoring tool and observe how JVM allocates Memory.

* **VisualVM** provides detailed information about Java applications while they are running on the Java Virutal Machine (JVM). VisualVM's graphical user interface enables us to quickly and easily see information about multiple Java applications.
* **Installing VisualVM**
  + Download the VisualVM installer from the VisualVM project page.
  + Extract the VisualVM installer to an empty directory on local system.
* **Starting VisualVM**
  + To start VisualVM on Windows, run the **visualvm.exe** program that is in the \bin folder under the VisualVM install folder.
  + After opening VisualVM, double click on VisualVM icon which appears on left panel.
* **Exploring VisualVM:**
  + Click on overview tab to know more about JVM arguments, System Properties etc.
  + Click on monitor tab to understand about CPU usage, Heap Space, Classes and Threads. Observe how memory is managed by JVM.
  + Click on Heap Dump(right-side corner) in monitor tab to know about classes, instances, and environment.
  + Click on threads to know how many types threads are running in live and finished threads.
  + Click on sampler to know CPU samples, Memory Samples.



# Program No.7: Explain memory allocation through Java programs.

**What is Stack Memory?**

* Stack in Java is a section of memory which contains methods, local variables, and reference variables. Stack memory is always referenced in Last-In-First-Out order. Local variables are created in the stack.

**What is Heap Memory?**

* Heap is a section of memory which contains Objects and may also contain reference variables. Instance variables are created in the heap.

**Memory Allocation in Java:**

* **Memory Allocation in Java** is the process in which the virtual memory sections are set aside in a program for storing the variables and instances of structures and classes. However, the memory isn’t allocated to an object at declaration but only a reference is created. For the memory allocation of the object, **new()** method is used, so the object is always allocated memory on the heap.
* The Java Memory Allocation is divided into following sections :

1. Heap
2. Stack
3. Code
4. Static

* This division of memory is required for its effective management.
  + The **code** section contains your **bytecode**.
  + The **Stack** section of memory contains **methods, local variables, and reference variables.**
  + The **Heap** section contains **Objects** (may also contain reference variables).
  + The **Static** section contains **Static data/methods**. **Difference between Local and Instance Variable**

**Instance variable** is declared **inside a class but not inside a method**

class Student

{

int num; // num is instance variable public void showData{}

}

**Local variables** are declared **inside** a **method including** method **arguments**. public void sum(int a)

{

int x = int a + 3; // a , x are local variables;

}

**Difference between Stack and Heap**

* Let’s take an example to understand this better.
* Consider that the main method calling method **m1**

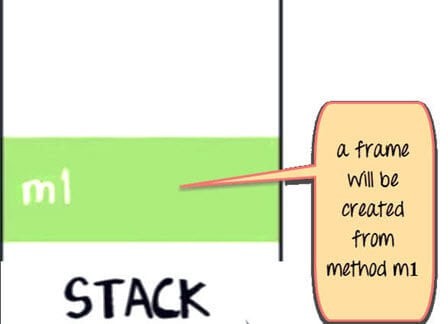
public void m1

{

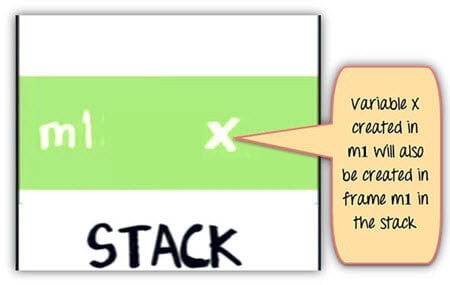
int x=20;

}

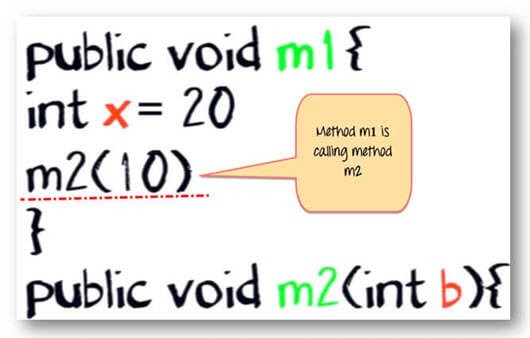
In the stack java, a frame will be created from method m1.

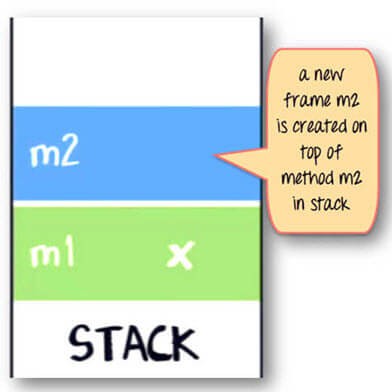


The variable **x** in m1 will also be created in the frame for m1 in the stack. (See image below).



As shown below Method m1 is calling method m2. In the stack Java, a new frame is created for m2 on top of the frame m1.



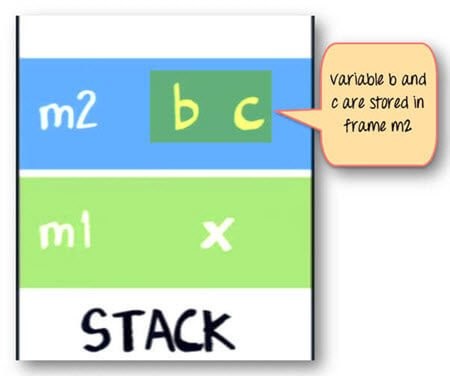


Variable b and c will also be created in a frame m2 in a stack. public void m2(int b)

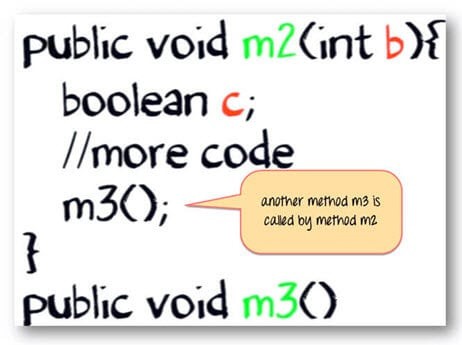
{

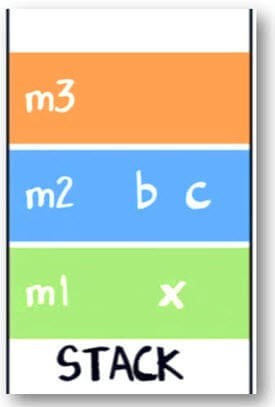
boolean c;

}



As shown below same method m2 is calling method m3. Again a frame m3 is created on the top of the stack (see image below).





Now let say our method m3 is creating an object for class **“Account,”** which **has two instances variables int p and int q.**

Account

{

int p; int q;

}

Here is the code for method m3

public void m3()

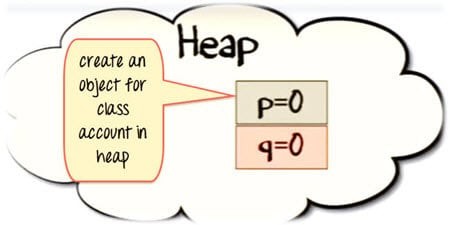
{

Account ref = new Account();

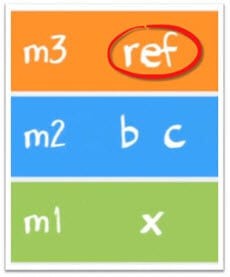
// more code

}

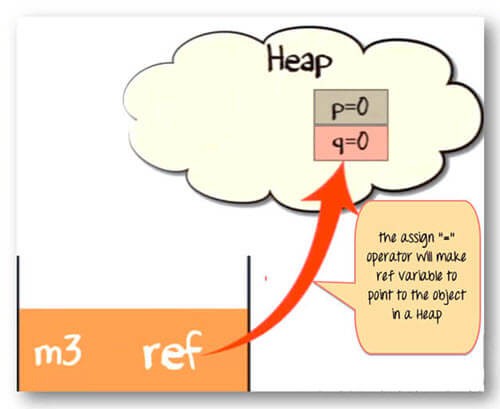
The statement new Account() will create an object of account in **heap.**

****

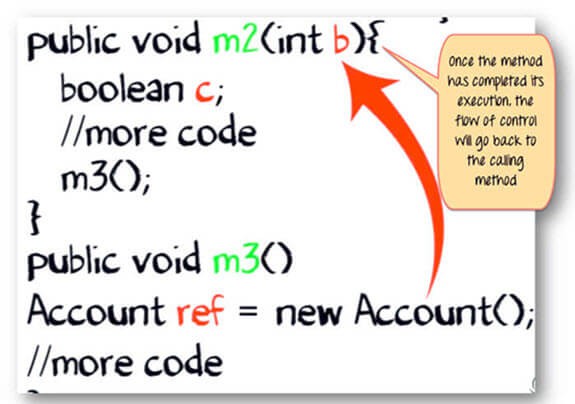
The reference variable “ref” will be created in a stack java.



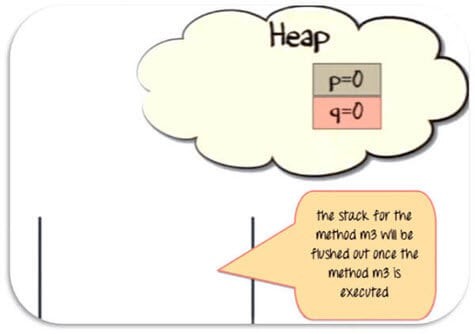
The assignment “=” operator will make a reference variable to point to the object in the Heap.



Once the method has completed its execution. The flow of control will go back to the calling method. Which in this case is method m2.



The stack from method m3 will be flushed out.



Since the reference variable will no longer be pointing to the object in the heap, it would be eligible for garbage collection.



* Once method m2 has completed its execution. It will be popped out of the stack, and all its variable will be flushed and no longer be available for use.
* Likewise for method m1.Eventually, the flow of control will return to the start point of the program. Which usually, is the “main” method.

**Summary:**

* When a method is called, a frame is created on the top of the stack.
* Once a method has completed execution, the flow of control returns to the calling method and its corresponding stack frame is flushed.
* Local variables are created in the stack.
* Instance variables are created in the heap & are part of the object they belong to.
* Reference variables are created in the stack.

# Program No.8: Code, execute and debug programs that uses different control statements. Identify and resolve issues in the given code snippet.

1. **Write a Program to check whether the given Integer is Odd or Even using if-else statement.**

import java.util.Scanner; class OddEven

{

public static void main(String[] args)

{

int n;

Scanner s = new Scanner(System.in); System.out.println("Enter the number you want to check:"); n = s.nextInt();

if(n % 2 == 0)

{

}

}

**Output:**

}

else

{

}

System.out.println("The given number "+n+" is Even ");

System.out.println("The given number "+n+" is Odd ");

**Enter the number you want to check:12 The given number 12 is Even**

**Enter the number you want to check:7 The given number 7 is Odd**

1. **Write a program to illustrate switch statement**

public class SwitchDemo

{

public static void main(String[] args)

{

int day = 4; switch (day)

{

case 1: System.out.println("Monday"); break;

case 2: System.out.println("Tuesday"); break;

case 3: System.out.println("Wednesday"); break;

case 4: System.out.println("Thursday"); break;

case 5: System.out.println("Friday"); break;

case 6: System.out.println("Saturday"); break;

case 7: System.out.println("Sunday"); break;

}

}

}

**Output:**

**Thursday**

1. **Write a Program to Generate n Fibonacci Numbers using for loop.**

import java.util.Scanner; class Fibonacci

{

public static void main(String[] args)

{

int n, a = 0, b = 0, c = 1;

Scanner s = new Scanner(System.in); System.out.println("Enter value of n:"); n = s.nextInt(); System.out.print("Fibonacci Series:"); for(int i = 1; i <= n; i++)

{

a = b;

b = c;

c = a + b; System.out.print(a+" ");

}

}

}

**Output:**

**Enter value of n:8**

**Fibonacci Series:0 1 1 2 3 5 8 13**

1. **Write a Program to Reverse a Number and Check if it is a Palindrome using while loop.**

import java.util.Scanner; class Palindrome

{

public static void main(String args[])

{

int n, m, a = 0,x;

Scanner s = new Scanner(System.in); System.out.println("Enter any number:"); n = s.nextInt();

m = n;

while(n > 0)

{

x = n % 10;

a = a \* 10 + x; n = n / 10;

}

if(a == m)

{

System.out.println("Given number "+m+" is Palindrome");

}

else

{

System.out.println("Given number "+m+" is Not Palindrome");

}

}

}

**Output:**

**Enter any number:565**

**Given number 565 is Palindrome**

**Enter any number:234**

**Given number 234 is Not Palindrome**

1. **Write a Program to Reverse a Number and find the Sum of its Digits using do-while Loop.**

import java.util.Scanner; class DoWhile

{

public static void main(String[] args)

{

int n, a, m = 0, sum = 0;

Scanner s = new Scanner(System.in); System.out.print("Enter any number:"); n = s.nextInt();

do

{

a = n % 10;

m = m \* 10 + a; sum = sum + a; n = n / 10;

}

}

**Output:**

while( n > 0); System.out.println("Reverse:"+m); System.out.println("Sum of digits:"+sum);

}

**Enter any number:35 Reverse:53**

**Sum of digits:8**

1. **Write a Program to Check whether the given Number is Prime Number. (Use break statement)**

import java.util.Scanner; class CheckPrime

{

public static void main(String args[])

{

int j, x, flag = 1; System.out.print("Enter any number :"); Scanner s = new Scanner(System.in);

x = s.nextInt();

for( j = 2; j < x; j++)

{

if(x % j == 0)

{

flag = 0; break;

}

}

if(flag == 1)

{

System.out.println("The "+x+" is a prime number.");

}

else

{

System.out.println("The "+x+" is not a prime number.");

}

}

}

**Output:**

**Enter any number :45**

**The 45 is not a prime number.**

**Enter any number :23 The 23 is a prime number.**

# Program No.9: Code, execute and debug programs that uses encapsulation concept.

class Encapsulate

{

// private variables declared these can only be accessed by public methods of class private String geekName;

private int geekRoll; private int geekAge;

public void setAge(int newAge) // set method for age to access private variable geekage

{

geekAge = newAge;

}

public void setName(String newName) // set method for name to access private variable geekName

{

geekName = newName;

}

public void setRoll(int newRoll) // set method for roll to access private variable geekRoll

{

geekRoll = newRoll;

}

public String getName() // get method for name to access private variable geekName

{

return geekName;

}

public int getRoll() // get method for roll to access private variable geekRoll

{

return geekRoll;

}

public int getAge() // get method for age to access private variable geekAge

{

return geekAge;

}

}

public class EncapsulationDemo

{

public static void main(String[] args)

{

Encapsulate obj = new Encapsulate(); // Creating object

// setting values of the variables using set methods

obj.setName("Harish"); obj.setAge(19); obj.setRoll(51);

// Displaying values of the variables using get methods System.out.println("Geek's name: " + obj.getName()); System.out.println("Geek's age: " + obj.getAge()); System.out.println("Geek's roll: " + obj.getRoll());

**// Direct access of geekRoll is not possible due to encapsulation**

**// System.out.println("Geek's roll: " + obj.geekName); // Not possible**

}

}

**Output:**

**Geek's name: Harish Geek's age: 19**

**Geek's roll: 51**

# Program No.10: Define class & implement like simple calculator and check compliance with SRP.

**import** java.util.Scanner;

**public class** BasicCalculator

{

**public static void** main(String[] args)

{

**double** num1; **double** num2; **double** ans; **char** op;

Scanner reader = **new** Scanner(System.*in*); System.*out*.println("Enter two numbers: "); num1 = reader.nextDouble();

num2 = reader.nextDouble(); System.*out*.println("\nEnter an operator (+, -, \*, /): "); op = reader.next().charAt(0);

**switch**(op) {

**case** '+': ans = num1 + num2;

**break**;

**case** '-': ans = num1 - num2;

**break**;

**case** '\*': ans = num1 \* num2;

**break**;

**case** '/': ans = num1 / num2;

**break**;

**default**: System.*out*.println("Error! Enter correct operator");

**return**;

}

System.*out*.println("\nThe result is given as follows:\n"); System.*out*.println(num1 + " " + op + " " + num2 + " = " + ans);

}

}

# Output:

**Enter two numbers:**

**12**

**78**

**Enter an operator (+, -, \*, /): \***

**The result is given as follows:**

**12.0 \* 78.0 = 936.0**