Dt : 17/9/2022

EX-program:

wap to to validate rollNo belongs to the branch or not?

GenerateBranch.java

```
package p1;
public class GenerateBranch {
     public String generate(String brCode)
{
        return switch(brCode) {
        case "01":yield "CIVIL";
        case "02":yield "EEE";
        case "03":yield "MECH";
        case "04":yield "ECE";
        case "05":yield "CSE";
        default:yield null;
        };
     }
}
```

CheckBranch.java

```
package p1;
public class CheckBranch {
   public boolean verify(String br)
   {
       return switch(br)
       {
            case "CIVIL":yield true;
            case "MECH":yield true;
            case "CSE" : yield true;
            case "ECE" : yield true;
            case "EEE" : yield true;
            default : yield false;
        };
    }
}
```

Percentage.java

```
package p1;
public class Percentage {
    public float calculate(int totMarks)
    {
        return
(float)totMarks/6;//TypeCasting
    }
}
```

StudentResult.java

```
package p1;
public class StudentResult {
    public String generate(float
per,boolean p)
    {
        if(p)
        {
            return "Fail";
        }
        else if(per>=70 && per<=100)
        {
            return "Distinction";
        }
        else if(per>=60 && per<70)
        {
            return "FirstClass";
        }
        else if(per>=50 && per<60)
        {
            return "SecondClass";
        }
        else if(per>=35 && per<50)
        {
            return "ThirdClass";
        }
        else
        {
            return "Fail";
        }
    }

}
```

StuMainClass.java(MainClass)

```
package p2;

import java.util.*;

import p1.*;

public class StuMainClass {

        public static void main(String[] args) {

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

```
System.out.println("Enter the rollNo:");
String rollNo = s.nextLine();
int len = rollNo.length();
if(len==10)
{
    String brCode = rollNo.substring(6,8);
    GenerateBranch gb = new GenerateBranch();
    String genBranch = gb.generate(brCode);
    if(genBranch==null)
    {
            System.out.println
            ("RollNO not matched with available branches");
    }//end of if
    else
    {
            System.out.println("Enter the name:");
            String name = s.nextLine();
            System.out.println("Enter the branch(CIVIL MECH EEE ECE CSE):");
            String br = s.nextLine().toUpperCase();
            CheckBranch cb = new CheckBranch();
            boolean k = cb.verify(br);
            if(k)
            {
                    if(genBranch.equals(br))
```

```
{
        System.out.println("===Enter six Sub marks====");
        int i=1;//Initialization
        int totM=0;
        boolean p=false;
        while(i<=6)
        {
                System.out.println("Enter Marks of Sub-"+i);
                int sub = s.nextInt();
                if(sub<0 || sub>100)
                {
                        System.out.println("Invalid marks...");
                        continue;
                        //skip the below lines from the Iteration
                }
                if(sub>=0 && sub<=34)//Fail Condition
                {
                        p=true;
                }
                totM=totM+sub;
                i++;
        }//end of loop
        System.out.println("TotMarks:"+totM);
        Percentage pr = new Percentage();
```

```
                              float per = pr.calculate(totM);

                              System.out.println("Per:"+per);

                              StudentResult sr = new StudentResult();

                              String result = sr.generate(per, p);

                              System.out.println("Result:"+result);

                         }//end of if

                         else

                         {

                              System.out.println("RollNo not matched with entered branch");

                         }

                    }//end of if

                    else

                    {

                         System.out.println("Invalid branch...");

                    }

               }

          }//end of if

          else

          {

            System.out.println("Invalid rollNO...");

          }

          s.close();

               }

     }
```

==

Program-1(Solution)

Define a method which returns the 1 if the given number is even,

in other case return 0

Name of method: isEven()

// which accepts an integer value as argument and return 1 if

the given number is even, else return 0.

Argument: int

Return type: an integer value

Example, if x = 22, return 1. if x = 35, return 0

Test.java

```
package maccess;
public class Test {
   public int isEven(int x)
   {
      if(x%2==0) return 1;
      else return 0;
   }
}
```

Program1.java(MainClass)

```
package maccess;
import java.util.*;
public class Program1 {
   public static void main(String[] args)
{
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the int 
value:");
        int x = s.nextInt();
        Test ob = new Test();
        int p = ob.isEven(x);
        System.out.println("Result : "+p);
        s.close();
    }
}
```

==

Program-2(Solution)

Define a method which returns the greatest number among two numbers.

Write the method with the following specifications:

Name of method: getGreatest()

// which accepts two integer values as argument and

return the greatest value.

Arguments: two argument of type integer

Return type: an integer value

Specifications: The value returned by the method getGreatest()

is determined by the following rules:

If any of the given numbers are negative, return -1.

If any of the given numbers are zero, return -2.

If the given numbers are positive, return the greatest.

Test.java

```
package maccess;
public class Test {
   public int getGreatest(int x,int y)
   {
        if(x<0 || y<0)
        {
           return -1;
        }
        else if(x==0 || y==0)
        {
           return -2;
        }
        else
        {
           if(x>y)
           {
              return x;
           }
           else
           {
              return y;
           }

        }
   }
}
```

Program2.java(Solution)

```
package maccess;
```

```
import java.util.*;
public class Program2 {
    public static void main(String[] args)
{
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the int
value of x:");
        int x = s.nextInt();
        System.out.println("Enter the int
value of y:");
        int y = s.nextInt();

        Test ob = new Test();
        int p = ob.getGreatest(x, y);
        System.out.println("Result : "+p);
        s.close();
    }
}
```

==

Program-3(Solution)

Define a method which returns the least number among two numbers.

Write the method with the following specifications:

Name of method: getLeastNum() // which accepts two integer values as argument and return the least value.

Arguments: two argument of type integer

Return type: an integer value

Specifications: The value returned by the method getLeastNum() is determined by the following rules:

If any of the given numbers are negative, return -1.

If any of the given numbers are zero, return -2.

If the given numbers are positive, return the least number.

Test.java

```
package maccess;
public class Test {
   public int getGreatest(int x,int y)
   {
       if(x<0 || y<0)
       {
          return -1;
       }
       else if(x==0 || y==0)
       {
          return -2;
       }
       else
       {
          if(x<y)
          {
             return x;
          }
          else
          {
             return y;
          }

       }
   }
}
```

Program3.java(Solution)

```
package maccess;
import java.util.*;
public class Program3 {
    public static void main(String[] args)
{
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the int
value of x:");
        int x = s.nextInt();
        System.out.println("Enter the int
value of y:");
        int y = s.nextInt();

        Test ob = new Test();
        int p = ob.getGreatest(x, y);
        System.out.println("Result : "+p);
        s.close();
    }
}
```

===

Program-4

Define a method which returns the number if it is an even number,

if the number is odd then return the next multiple of 10.

Write the method with the following specifications:

Name of method: oddRounder()

// which accepts an integer value as argument and

return the same value if it is an even number,

if the value is odd then return the next multiple of 10.

Arguments: one argument of type integer

Return Type: an integer value

Example if x = 24 then return 24, if x = 25 then return 30.

Specifications: The value returned by the method oddRounder()

is determined by the following rules:

If any of the given number is negative, return -1.

If any of the given number is zero, return -2.

If the given number is even, return the same number.

If the given number is odd, return the next multiple of 10.

Test.java

```
package maccess;
public class Test {
   public int oddRounder(int x)
   {
       if(x<0)
       {
          return -1;
       }
       else if(x==0 )
       {
          return -2;
       }
```

```
        else
        {
               if(x%2==0)
               {
                return x;
               }
               else
               {
                int k = x/10;
                return ((k+1)*10);
               }

        }
    }
}
```

Program4.java(Solution)

```
package maccess;
import java.util.*;
public class Program4 {
    public static void main(String[] args)
{
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the int
value of x:");
        int x = s.nextInt();

        Test ob = new Test();
        int p = ob.oddRounder(x);
        System.out.println("Result : "+p);
        s.close();
```

```
          }
}
```

===

Program-5(Solution)

Define a method which returns the 1 if the given number is positive,

return -1 if the given number is negative,

return 0 if the given number is 0.

Name of method findSign()

Arguments: one argument of type integer

Return Type: an integer value

Test Cases

1. ***If any of the given number is positive, return 1.***
2. ***If any of the given number is negative, return -1.***
3. ***If any of the given number is zero, return 0.***

Dt : 19/9/2022

(c)Using 'is equal to'(==) operator:

=>'is equal to'(==) operator will compare the references of

Objects,which means it will compare the content of objects.

Ex : DemoStrin10.java

```
package maccess;
public class DemoString10 {
      public static void main(String[] args) {
              //String Literal Process
            String s1 = "nit";
            String s2 = "nit";
            System.out.println("====String Literal process====");
```

```
          if(s1==s2) {
               System.out.println("Strings are Equal....");
          }else {
               System.out.println("Strings are Not-Equal...");
          }
               //new operator Process
          String s3 = new String("hyd");
          String s4 = new String("hyd");
          System.out.println("====new operator process====");
          if(s3==s4) {
               System.out.println("Strings are Equal....");
          }else {
               System.out.println("Strings are Not-Equal...");
          }
               //String literal and new Operator process
          String s5 = "java";
          String s6 = new String("java");
          System.out.println("====String literal and new
operator process====");
          if(s5==s6) {
               System.out.println("Strings are Equal....");
          }else {
               System.out.println("Strings are Not-Equal...");
          }
     }
}
```

o/p:

```
====String Literal process====
Strings are Equal....
====new operator process====
Strings are Not-Equal...
====String literal and new operator process====
Strings are Not-Equal...
```

----

faq:

define String Constant pool?

=>The separate partition of HeapArea where String objects are created is known as String Constant Pool.

=>This String Constant pool will restrict duplicate String objects creation.

faq:

wt is the diff b/w 'String literal process' and 'new operator process' in creating String objects?

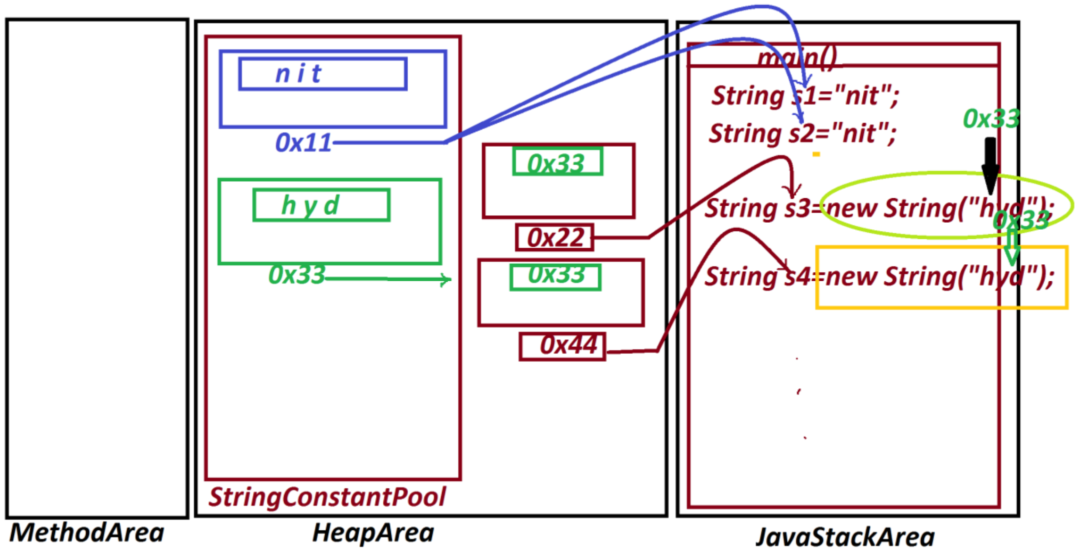
(i)In String Literal process,the execution engine will check the String Constant pool is any object having the same data,

=>If Object is not available then new object is created.

=>If Object is available then reference of existing object is is used without creating new Object.

(ii)In new operator process,one object is created directly in HeapArea and the object will hold the reference of Object created in String Constant Pool.

Diagram:



===

Summary:

=>'is equal to' operator is not preferable to use on Non-Primitive datatypes,because it compares Object references and generate Wrong results.

=>But,'is equal to' operator can be used on Primitive datatypes.

===

****imp***

2.java.lang.StringBuffer class:

=>The Objects which are created using 'java.lang.StringBuffer' class or mutable objects.(Object once created can be modified are known as Mutable objects)

=>The following are four constructors from 'StringBuffer' class:

public java.lang.StringBuffer();

public java.lang.StringBuffer(int);

public java.lang.StringBuffer(java.lang.String);

public java.lang.StringBuffer(java.lang.CharSequence);