26-6 - Strings

P6-10

```
import java.util.Scanner;
class StringDemo2
{
public static void main(String args[])
{
//6
System.out.println("Bmsce equals Bmsce = "+
                   ("Bmsce".equals("Bmsce")));
System.out.println("Bmsce equals college = "+
                   ("Bmsce".equals("College")));
System.out.println("Bmsce equals BMSCE = "+
                   ("Bmsce".equals("BMSCE")));

//7
String S1 = "Bmsce college";
String S2 = "Welcome to Bmsce College of Engineering";
if(S1.regionMatches(0,S2,0,13))
System.out.println("substring matched");
else
System.out.println("not matched");

//8
System.out.println("startsWith() demo:");
System.out.println("Does eclipse start with
                   ecl?:"+("eclipse".startsWith("ecl")));
System.out.println("Does eclipse start with
                   'ec?:"+("eclipse".endsWith("tec")));
```

```
//9
System.out.println("endsWith() demo:");
System.out.println("Does eclipse end with
                    pse?:" + ("eclipse".endsWith
                    ("pse")));
System.out.println("Does eclipse end with ecl?:
                    +("eclipse".endsWith("ecl")));

//10
System.out.println("equals versus ==");
String o="Beautiful";
String p="Beautiful";
System.out.println("String 1:"+o);
System.out.println("String 2:"+p);
System.out.println("Equals:"+(o.equals(p)));
System.out.println("==" + (o==p));
}
}
```

Output:

```
Bmsce equals Bmsce = true
Bmsce equals College = false
Bmsce equals BMSCE = false
Not matched
```

startsWith demo():

Does eclipse start with ecl?: true

Does eclipse ~~end~~ start with lec?: false

endsWith demo():

Does eclipse ~~start~~ end with pse?: true

Does eclipse end with ecl?: false

equals versus ==

String 1: Beautiful

String 2: Beautiful

Equals: true

== true

~~Intro~~

Q. 19

Abstract class Fly

Methods fly(), makesound()

Subclasses - Eagle, Hawk

```
abstract class Fly
{
  public abstract void fly();
  public abstract void makesound();
}

class Eagle extends Fly
{
  public void fly()
  {
    System.out.println("The eagle flies high.");
  }
```

```
public void makesound()
 {
    System.out.println("The eagle emits a high pitched whistle.");
 }
}

class Hawk extends Fly
 {
  public void fly()
  {
    System.out.println("The hawk flies lower");
  }
  public void makesound()
  {
    System.out.println("The hawk screeches keaaar");
  }
}

class Birdmain
 {
  public static void main(String args[])
  {
    System.out.println("Invoking eagle!");
    Eagle eg=new Eagle();
    eg.fly();
    eg.makesound();
```

```
System.out.println("Invoking hawk!");
Hawk hk = new Hawk();
hk.fly();
hk.makesound();
}
}
```

Output:

Invoking eagle!
The eagle flies high
The eagle emits a high pitched whistle.
Invoking hawk!
The hawk flies lower
The hawk screeches keaaar

21. Implementation of stack using Generics:

```
public class Stack<E>
{
  E stck[];
  int top;
  final int SIZE = 10;
  Stack()
  {
   stck = (E[]) new Object[SIZE];
   top = -1;
}
```

```
void push (E item)
{
  if (top == SIZE-1)
  System.out.println ("Stack is full");
  else
  stck [++top] = item;
  }

E pop ()
{
  if(top < 0)
  {
    System.out.println("Stack underflow");
    return null;
    }
  else
  return stck [top--];
  }
 }

import java.util.Scanner;
public class TestStack
 {
  public static void main (String args[])
  {
```

```
Stack <Integer> mystack1 = new Stack
Stack <Integer>();

Stack <Double> mystack2 = new Stack <Double >();
Scanner s = new Scanner (System.in);
System.out.println ("Enter elements into integer stack:");
for(int i=0; i<5; i++)
{ int n=s.nextInt();
  mystack1.push (n);
}
System.out.println ("Enter elements into the double
                     stack:");
for(int i=0; i<5; i++)
{
  double m=s.nextDouble();
  mystack2.push(m);
}
System.out.println ("Elements of stack 1");
for(int i=0; i<5; i++)
{
  System.out.println (mystack1.pop());
}
System.out.println ("Elements of stack 2");
for(int i=0; i<5; i++)
{
```

```
System.out.println(mystack2.pop());
}
s.close();
}
}
```

Output:

```
Enter elements into the integer stack:
1 2 3 4 5
Enter elements into the double stack:
8.0 9.0 10.5 11.6 12.7

Elements of stack 1:
5
4
3
2
1

Elements of stack 2:
12.7
11.6
10.5
9.0
8.0
```

16/11/24