16/1/24

Lab - 06.

1. Demonstrate various string constructor with proper java programs.

```
class string
{
  public static void main (string args[])
  {
    char c[] = {'J', 'a', 'v', 'a'};
    String s1 = new String (c);
    String s2 = new String (s1);
    System.out.println (s1);
    System.out.println (s2);
  }
}
```

O/P:

JAVA
JAVA

2. Demonstrate startsWith() to give output true and false.

```
public class Main
{
  public static void main (String[] args)
  {
    String test = "testString";
    String pattern = "te";
    System.out.println (test.startsWith (pattern));
```

```
    pattern = "el";
    System.out.println(test.startsWith(pattern));
}
```

O/P: True
false.

19. Write a Java prgm to create an abstract class Bird with abstract methods fly() and makeSound(). Create subclasses Eagle and Hawk that extend the Bird class and implement the respective methods to describe how each bird flies and makes a sound.

```
abstract class Bird
{
    abstract void fly();
    abstract void makeSound();
class Eagle extends Bird
{
    void fly()
    {
        System.out.println("Eagle is flying high");
    }
void makeSound()
{
    System.out.println("Eagle is making a screechy sound");
}
};
```

```
class Hawk extends Bird
{
  void fly()
  {
    System.out.println("Hawk is flying with a
        speed");
  }
void makeSound()
  {
    System.out.println("Hawk is making a
      crying sound");
  }
}
public class Main {
  public static void main(String[] args)
  {
    Eagle eagle = new Eagle();
    Hawk hawk = new Hawk();
    eagle.fly();
    eagle.makeSound();
    hawk.fly();
    hawk.makeSound();
  }
}
```

O/P:, Eagle is flying high.
Eagle is making screeching sound
Hawk is flying with a speed
Hawk is making a crying sound.

## Generics

1. Write a java prgm to create a generic class Stack which hold 5 integers and 5 double values.

```
import java.util.EmptyStackException;
public class Stack <T>
{
    private int maxsize;
    private int top;
    private Object[] stackArray;

    public Stack(int size)
    {
        maxsize = size;
        stackArray = new Object[maxsize];
        top = -1;
    }

    public void push(T value)
    {
        if(top < maxsize-1)
        {
            top++;
            stackArray[top] = value;
        }
        else
        {
            throw new RuntimeException("stack is full");
        }
    }
```

```
public T pop()
{
  if(!isEmpty())
  {
    return (T)stackArray[top--];
  }
  else{
    throw new EmptyStackException();
  }
}
public boolean isEmpty()
{
  return (top == -1);
}
public int size()
{
  return top+1;
}
public static void main(String[] args)
{
  Stack<Integer> intStack = new Stack<>(5);
  Stack<Double> DoubleStack = new Stack<>(5);
  for(int i = 0; i<5; i++)
  {
    intStack.push(i);
    doubleStack.push((double)i);
  }
  System.out.println("Integer Stack:");
  for(i = 0; i<5; i++)
  {
    System.out.println(intStack.pop());
  }
```

```
System.out.println("Double Stack");
for(i=0; i<5; i++)
{
    sop (doubleStack.pop());
}
}
```

o/p; Integer Stack:

4
3
2
1
0

Double Stack:

4.0
8.0
2.0
1.0
0.0

16/01/24