1. This Question based on Collections and generics.(10 marks)

Write a program where it prints the sentence words in the reverse order using the stack class. Get the sentence word by word as a keyboard input until user enter a word with a period. Once a word with period finds, it should print the reverse order of that sentence.

You can Refer the main class and the console output given below and implement the PrintReverse class

```
2
3  import java.util.Scanner;
4  import java.util.Stack;
5
6  public class Demo {
7
8      public static void main(String[] args) {
9          Stack<String> theStack = new Stack<String>();
10         PrintReverse ob=new PrintReverse();
11         ob.inputSentence(theStack);
12         ob.printSentence(theStack);
13
14     }
15
16 }
```

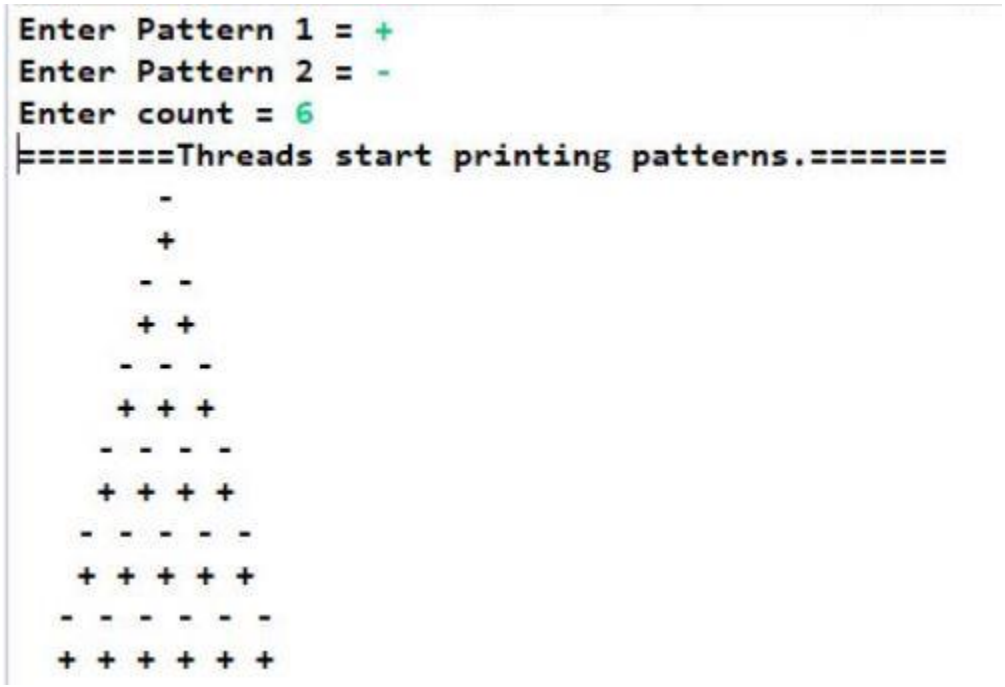
```
Enter the first word in your setence
i
Enter a word by word
like
Enter a word by word
java.
Reverse sentence
java.
like
i
```

2. This Question is based on Java Threads. (10 marks)

A Wrapping paper Art is printed using computer program and which is drawn using two concurrent Threads. You are allowed to enter pattern styles through keyboard inputs and you should select number of occurrences (count) to be printed the style. Each thread should print patterns one after the other and you should print the triangle shape using given style.

Refer the console output and implement the two threads

Output:



3. This question is based on design patterns.(10 marks)

In the Eclipse integrated development environment (IDE) software, we can perform some actions to support "save" and "saveall" actions. When Developer press the relevant button, IDE it should perform the different actions.

1. Write the most suitable design pattern name that can be use in above scenario?

2. Implement the relevant classes to demonstrate the design pattern you named in part 1.

You can refer to the main class given below and implement the classes accordingly.

```
1
2 public class Developer {
3     public static void main(String args[]) {
4         IDE ide=new IDE();
5         Receiver intendedreceiver = new Receiver();
6         System.out.println("Pressed save button!");
7         SaveAction saveaction = new SaveAction(intendedreceiver);
8         ide.SetAction(saveaction);
9         ide.ExecuteAction();
10
11        System.out.println("Pressed saveAll button!");
12        SaveAllAction saveallaction = new SaveAllAction(intendedreceiver);
13        ide.SetAction(saveallaction);
14        ide.ExecuteAction();
15    }
16
17 }
```

```
<terminated> Developer [Java Application] C:\Pr
Pressed save button!
Saving file...
Pressed saveAll button!
Saving all the files...
```

This question is based on the **Object-Oriented Programming (OOP) concepts**. You are going to implement a code for a "MenInblack" video game. There is an Alien class to represent a monster and an AlienPack class that represents a pack including different types of aliens. MenInBlack class has different types of agents who is going to kill the aliens and obtain the scores.

a) You can refer the output given in **Main** class and adjust your code accordingly.

```
public class Main {

    public static void main(String args[]) {

        //creating a alien pack with 5 different alians
        AlienPack pack1 = new AlienPack(5);
        pack1.addAlien(new MarshmalloAlien(), 0);
        pack1.addAlien(new OgreAlien(), 1);
        pack1.addAlien(new OgreAlien(), 2);
        pack1.addAlien(new SnakeAlien(), 3);
        pack1.addAlien(new MarshmalloAlien(), 4);

        MenInBlack AgentK = new MenInBlack(pack1);
        AgentK.kill();
        System.out.println("Your score is " + AgentK.getScore());
    }
}
```

```
put - Final2020 (run)  ⊠ | Test Results
run:
Your score is 55
BUILD SUCCESSFUL (total time: 0 seconds)
```

i). Implement the **Alien** interface and declare **getScore()** method.

ii). Create three classes called **MarshmalloAlien**, **OgreAlien** and **SnakeAlien** and implement the **Alien** interface in each class. MarshmalloAlien has 15, OgreAlien has 10 and SnakeAlien has 5 as their scores.

iii). Similarly create a class called **AlienPack** and implement the property of **aliens array (Alien[]).**

iv). Implement the **constructor** in the AlienPack class. Distinguish the number of aliens.

v). Implement the method called **addAlien** which accept a specific alien and the index.

vi). Implement the method called **getAliens()** which returns the alien array.

vii). Implement the **MenInBlack** class and implement the property of **score** and **alienpack(AlienPack).**

viii). Implement the **constructor** in the MenInBlack class, which accept and initialize the alienpack. Make the starting agent score as zero.

ix). Implement the getters and setters for score and alienpack.

x). Implement the **kill()** method which kills the aliens in the alienpack and calculate the total score depending on each alien.

b) "MenInblack" video game has another level where while the agent is killing the aliens, alien will do a damage for the agent and he will reduce the score accordingly.

i). Implement the **MenInBlackLevel2** class which is a child of the **MenInBlack** class.

ii). Implement the **constructor** in the **MenInBlackLevel2** class, which accept and initialize the alienpack.

iii). Override the kill method where it calculate the total score depending on each alien. Additionally, generate a random number for each alien. Each time the random number is an ODD, the total score will be reduce by 2.
*Hint: Math.random() *10 will generate a random decimal number between 1 and 10.*