INTRODUCTION TO JAVA 2

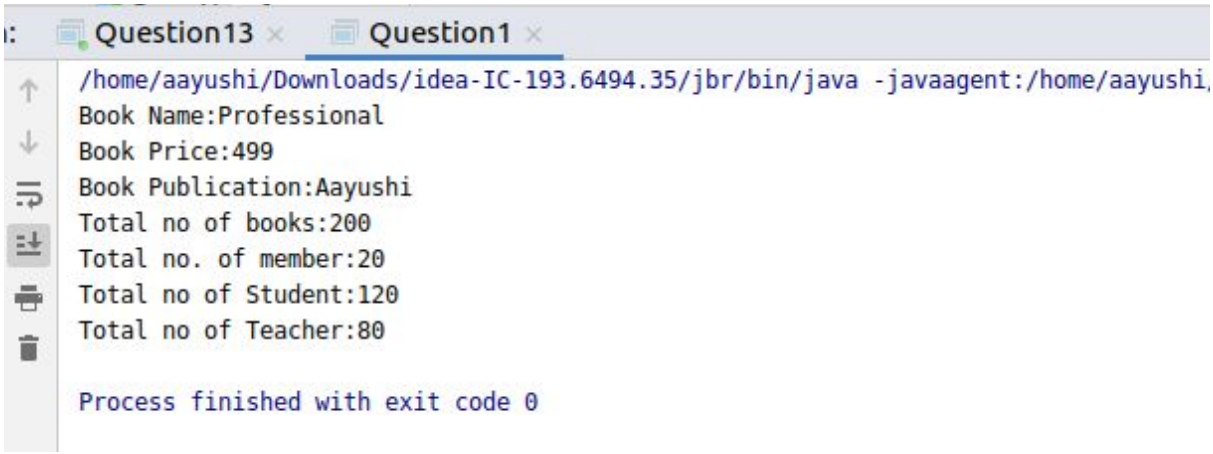
1. Create Java classes having suitable attributes for Library management system.Use OOPs concepts in your design.Also try to use interfaces and abstract classes.

CODE

```
package aayushi;
interface bookInfo{
   public void bookName(String Name);
   public void bookPrice(int price);
   public void bookPublication(String Publisher);
}
interface LibraryInfo{
   public void bookCount(int count);
   public void memberCount(int  memCount);
}
interface memberInfo{
   public void studentCount(int stCount);
   public void teacherCount(int tCount);
}
class Book implements bookInfo{

   @Override
   public void bookName(String Name) {
      System.out.println("Book Name:"+Name);
   }

   @Override
   public void bookPrice(int price) {
      System.out.println("Book Price:"+price);
   }

   @Override
   public void bookPublication(String Publisher) {
      System.out.println("Book Publication:" +Publisher);

   }
}
class Library implements LibraryInfo{

   @Override
   public void bookCount(int count) {
      System.out.println("Total no of books:"+count);
```

```
    }

    @Override
    public void memberCount(int memCount) {
        System.out.println("Total no. of member:"+memCount);
    }
}
class Member implements memberInfo{

    @Override
    public void studentCount(int stCount) {
        System.out.println("Total no of Student:"+stCount);
    }

    @Override
    public void teacherCount(int tCount) {
        System.out.println("Total no of Teacher:"+tCount);
    }
}
public class Question1 {
    public static void main(String[] args) {
        Book b= new Book();
        Library l= new Library();
        Member m=new Member();
        b.bookName("Professional");
        b.bookPrice(499);
        b.bookPublication("Aayushi");
        l.bookCount(200);
        l.memberCount(20);
        m.studentCount(120);
        m.teacherCount(80);
    }
}
```

OUTPUT

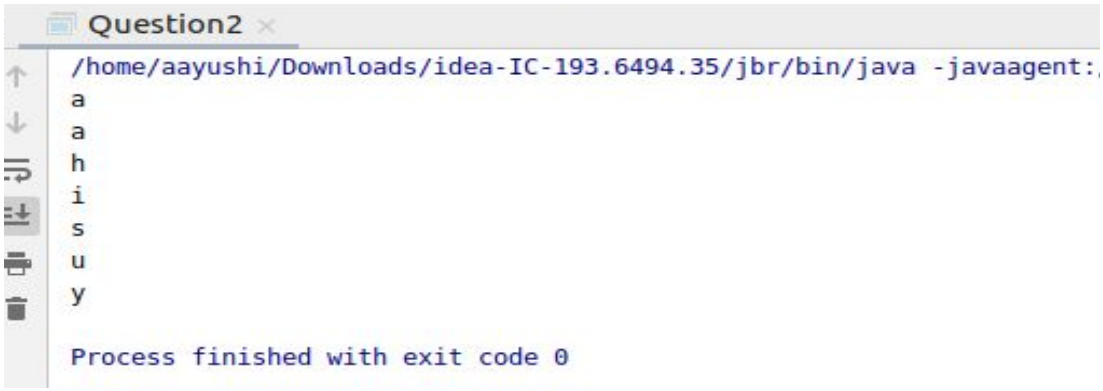


2. WAP to sorting string without using string Methods?

CODE

```
package aayushi;
public class Question2 {
  public static void main(String[] args) {
     String str = "aayushi";
     int j = 0;
     char temp = 0;
     char[] chars = str.toCharArray();
     for (int i = 0; i < chars.length; i++)
     {
        for (j = 0; j < chars.length; j++)
        {
           if (chars[j] > chars[i])
           {
              temp = chars[i];
              chars[i] = chars[j];
              chars[j] = temp;
           }
        }
     }
     for (int i = 0; i < chars.length; i++) {
        System.out.println(chars[i]);
     }
  }
}
```

OUTPUT



3. WAP to produce NoClassDefFoundError and ClassNotFoundException exception.

A.ClassNotFoundException : ClassNotFoundException occurs when you try to load a class at runtime using Class.forName() or loadClass() methods and requested classes are not found in classpath.

CODE

```
package aayushi;
public class Question3 {
  public static void main(String[] args) {
        try
        {
           Class.forName("Aayushi_test");
        }
        catch (ClassNotFoundException ex)
        {
           ex.printStackTrace();
        }
    }
}
```

OUTPUT

B. NoClassDefFoundError :NoClassDefFoundError occurs when class was present during compile time and program was compiled and linked successfully but class was not present during runtime.

CODE

```
package aayushi;
// Java program to illustrate
// NoClassDefFoundError

class ExceptionExample {
  public void print(){
     System.out.println("Hello");
  }
}

public class Question3 {
  public static void main(String[] args) {
     ExceptionExample e = new ExceptionExample();
     e.print();
     }
}
```

OUTPUT

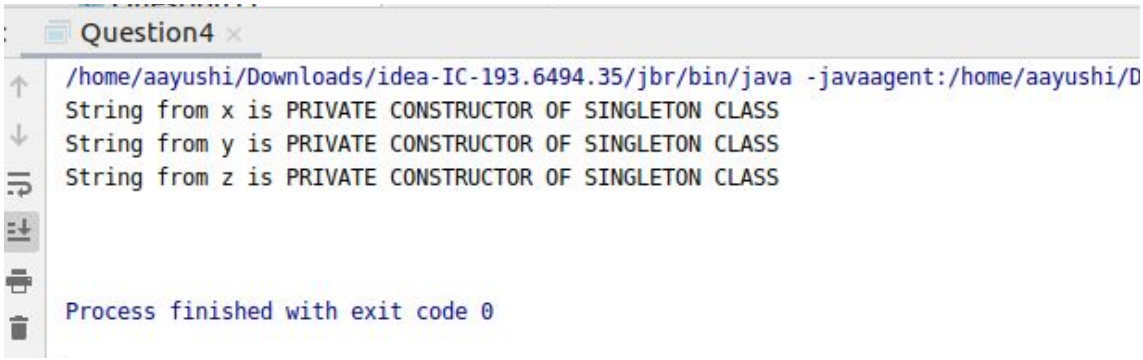
4. WAP to create a singleton class.

CODE

```
package aayushi;
class Singleton
{
  private static Singleton single_instance = null;       // static variable single_instance of type
Singleton
  public String s;

  private Singleton()       // private constructor restricted to this class itself
  {
     s = "Private constructor of Singleton class";
```

```
  }

  public static Singleton getInstance()       // static method to create instance of Singleton class
  {
     if (single_instance == null)
        single_instance = new Singleton();

     return single_instance;
  }
}

public class Question4 {
  public static void main(String[] args) {
     Singleton x = Singleton.getInstance();          // instantiating Singleton class with variable x
     Singleton y = Singleton.getInstance();
     Singleton z = Singleton.getInstance();

     x.s = (x.s).toUpperCase();              // changing variable of instance x

     System.out.println("String from x is " + x.s);
     System.out.println("String from y is " + y.s);
     System.out.println("String from z is " + z.s);
     System.out.println("\n");
  }
}
```

OUTPUT



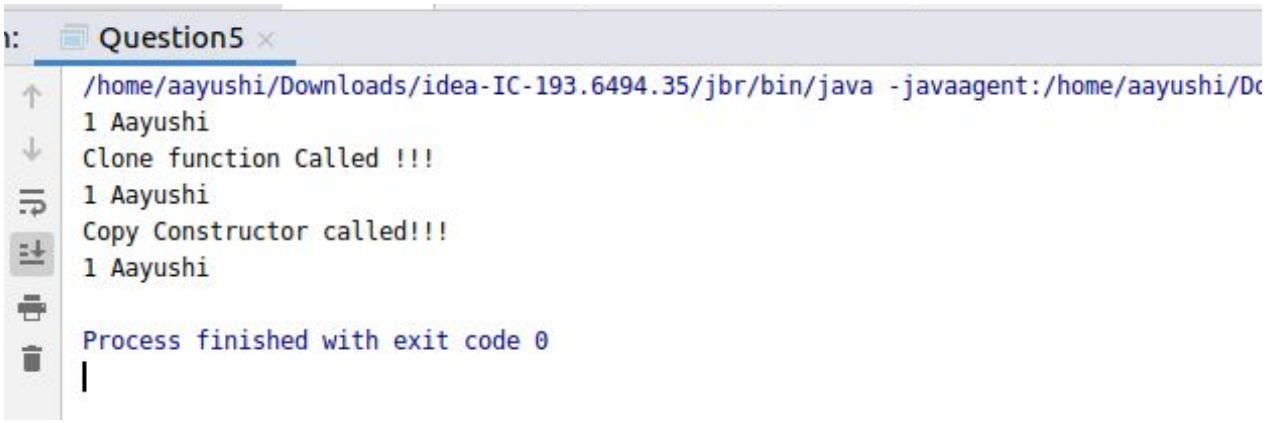
5. WAP to show object cloning in java using cloneable and copy constructor both.

CODE

```
package aayushi;
import java.lang.Cloneable;
//WAP to show object cloning in java using cloneable and copy constructor both.

public class Question5 implements Cloneable{
  int number;
  String str;

  Question5(int number , String str){
     this.number = number;
     this.str= str;
  }

  Question5(Question5 obj){
     System.out.println("Copy Constructor called!!!");
     number= obj.number;
     str= obj.str;

  }

  public Object clone()throws CloneNotSupportedException{
     System.out.println("Clone function Called !!!");

     return super.clone();
  }

  public static void main(String[] args) throws CloneNotSupportedException {
     try{


        Question5 originalObject1 = new Question5(1,"Aayushi");
        System.out.println(originalObject1.number+" " + originalObject1.str);

        Question5 cloneableObject2 = (Question5) originalObject1.clone();
        System.out.println(cloneableObject2.number+" " + cloneableObject2.str);

        Question5 copyConstructorObject3 =new Question5(originalObject1);
        System.out.println(copyConstructorObject3.number+" " + copyConstructorObject3.str);

     }
     catch (CloneNotSupportedException cnse){}
```

```
  }
}
```

OUTPUT



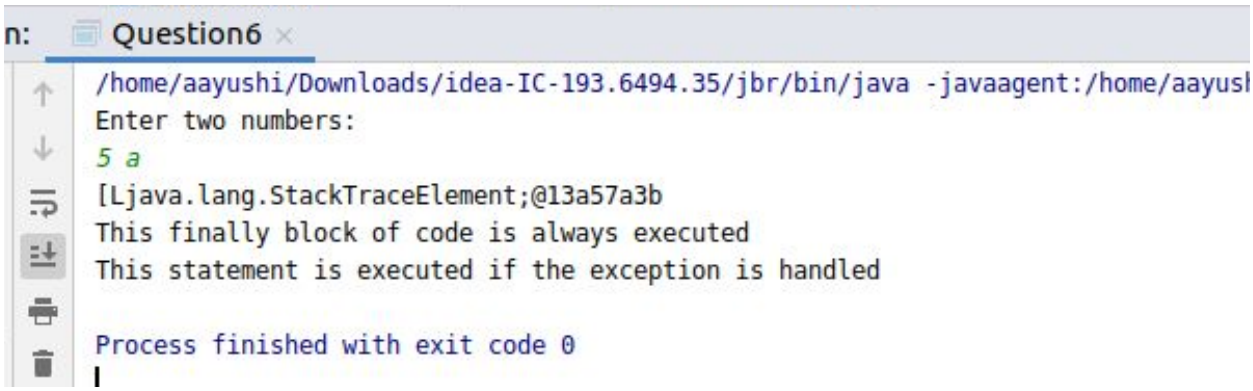
6. WAP showing try, multi-catch and finally blocks.

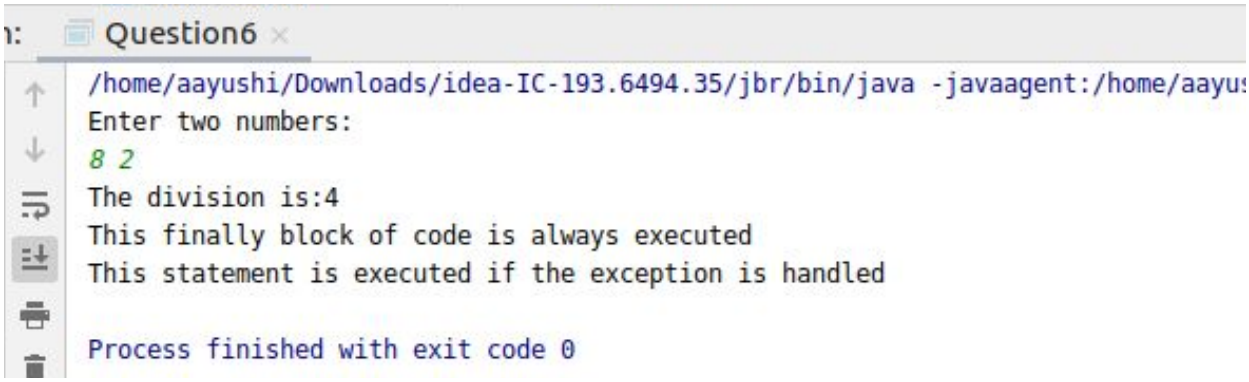
CODE

```
package aayushi;
import java.util.Scanner;
//6. WAP showing try, multi-catch and finally blocks.
public class Question6 {
  public static void main(String[] args) {
      System.out.println("Enter two numbers:");
      Scanner sc = new Scanner(System.in);
      try
      {
         int n = sc.nextInt();
         int n1 = sc.nextInt();
         int res = n/n1;
         System.out.println("The division is:"+res);
      }
      catch(ArithmeticException ex)
      {
         System.out.println("Arithmetic Exception"); // This exception occurs when an integer is
divided by zero.
      }
      catch(NumberFormatException ex) //unable to format (convert) a string into a number.
      {
         System.out.println("Number Format Exception");
      }
      catch(Exception  ex)
```

```
        {
            System.out.println(ex.getStackTrace());
        }
        finally {
            System.out.println("This finally block of code is always executed");
        }
        System.out.println("This statement is executed if the exception is handled");
    }
}
```

OUTPUT





7. WAP to convert seconds into days, hours, minutes and seconds.

CODE

```
package aayushi;
import java.util.Scanner;
//7. WAP to convert seconds into days, hours, minutes and seconds.
public class Question7 {
   public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the seconds: ");
        int n = sc.nextInt();

        int days = n / (24*3600);
        n = n % (24*3600);
        int hour = n / 3600;
        n = n % 3600;
        int min = n / 60;
        n = n % 60;

        System.out.println("The days:"+days);
        System.out.println("The hours:"+hour);
        System.out.println("The minutes:"+min);
        System.out.println("The seconds:"+n);
    }
}
```

OUTPUT

8. WAP to read words from the keyboard until the word done is entered. For each word except done, report whether its first character is equal to its last character. For the required loop, use a

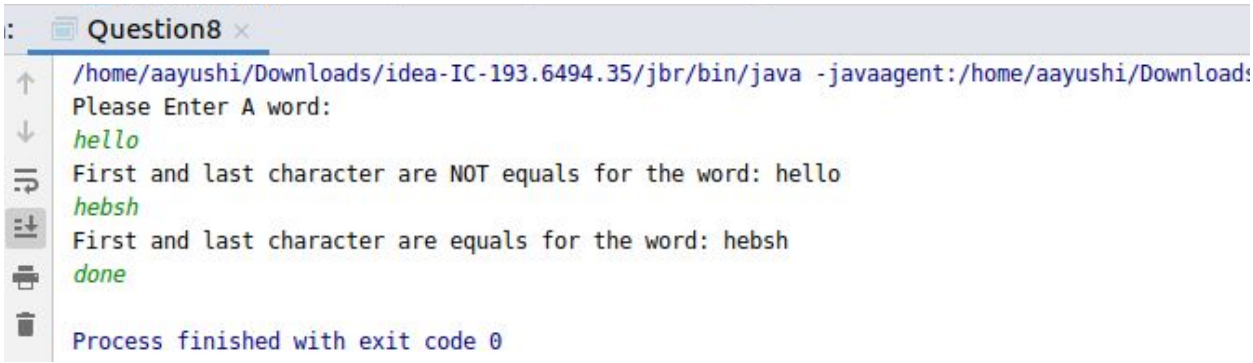
a)while statement

CODE

```
package aayushi;
import java.util.Scanner;

public class Question8 {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Please Enter A word: ");
        String word = keyboard.next();
        while(!word.equals("done"))
```

```
    {
        if(word.charAt(0) == word.charAt(word.length() - 1))
            System.out.println("First and last character are equals for the word: " + word);

        else
            System.out.println("First and last character are NOT equals for the word: " + word);

    word = keyboard.next();
    }
  }
}
```

OUTPUT



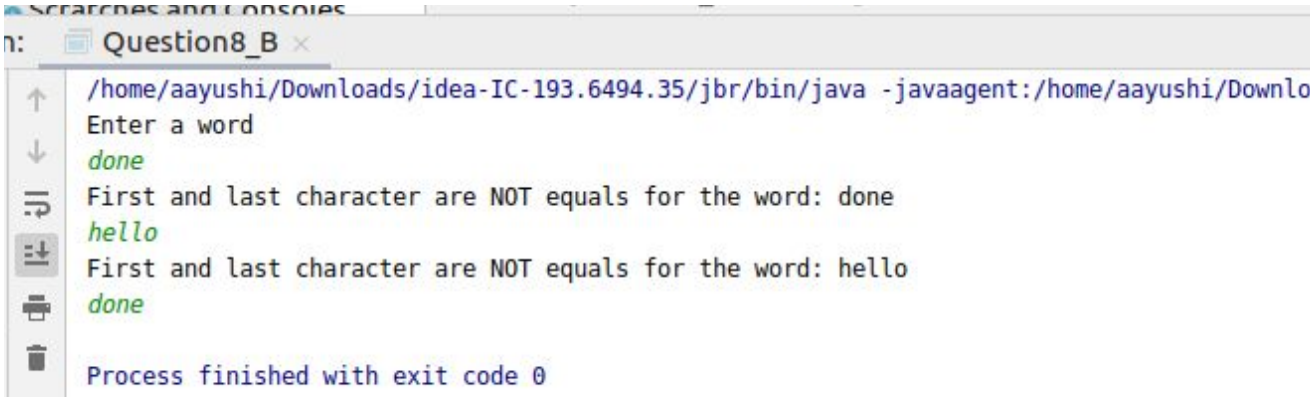
b)do-while statement

CODE

```
package aayushi;
import java.util.Scanner;

public class Question8_B {
  public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    System.out.println("Enter a word");
    String word = keyboard.next();
    do
    {
        if(word.charAt(0) == word.charAt(word.length() - 1))
        {
            System.out.println("First and last character are equals for the word: " + word);
        }
        else
```

```
        {
            System.out.println("First and last character are NOT equals for the word: " + word);
        }
        word = keyboard.next(); }
    while(!word.equals("done"));
  }
}
```

OUTPUT



9. Design classes having attributes for furniture where there are wooden chairs and tables, metal chairs and tables. There are stress and fire tests for each products.

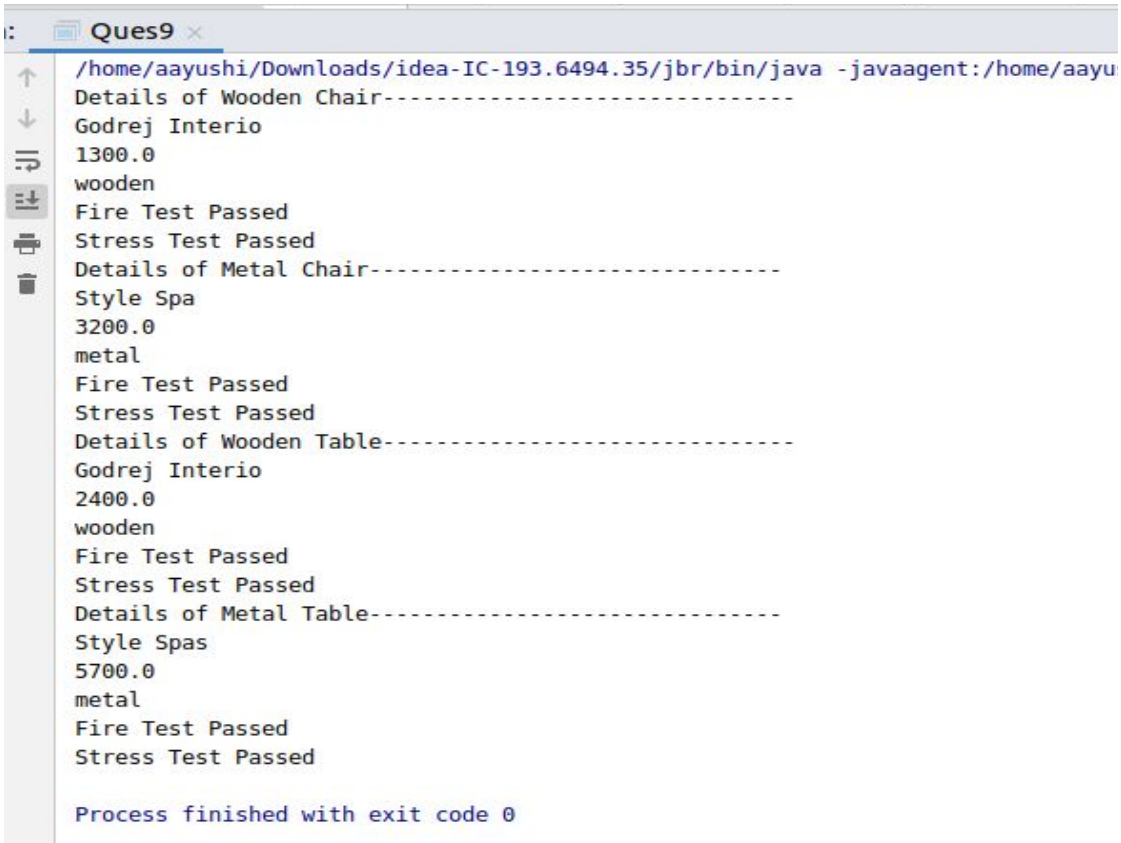
CODE

```
package aayushi;
public class Ques9 {
  public static void main(String[] args) {

      Chair woodenChair = new Chair("Godrej Interio",1300,"wooden");
      Chair metalChair = new Chair("Style Spa",3200,"metal");

      Table woodenTable = new Table("Godrej Interio",2400,"wooden");
      Table metalTable = new Table("Style Spas",5700,"metal");

      System.out.println("Details of Wooden Chair------------------------------");
      System.out.println(woodenChair.manufacturer);
      System.out.println(woodenChair.price);
      System.out.println(woodenChair.type);
      System.out.println(woodenChair.fireTest());
```

```
        System.out.println(woodenChair.stessTest());

        System.out.println("Details of Metal Chair-------------------------------");
        System.out.println(metalChair.manufacturer);
        System.out.println(metalChair.price);
        System.out.println(metalChair.type);
        System.out.println(metalChair.fireTest());
        System.out.println(metalChair.stessTest());

        System.out.println("Details of Wooden Table-------------------------------");
        System.out.println(woodenTable.manufacturer);
        System.out.println(woodenTable.price);
        System.out.println(woodenTable.type);
        System.out.println(woodenTable.fireTest());
        System.out.println(woodenTable.stessTest());

        System.out.println("Details of Metal Table-------------------------------");
        System.out.println(metalTable.manufacturer);
        System.out.println(metalTable.price);
        System.out.println(metalTable.type);
        System.out.println(metalTable.fireTest());
        System.out.println(metalTable.stessTest());
    }

}

class Furniture{
    String manufacturer;
    float price;

    Furniture(String manufacturer, float price){
        this.manufacturer = manufacturer;
        this.price = price;
    }

    public String stessTest(){

        return "Stress Test Passed";
    }
    public String fireTest(){

        return "Fire Test Passed";
    }
```

```
}

class Chair extends Furniture{
   String type ;

   Chair(String manufacturer, float price, String type){
      super(manufacturer,price);
      this.type = type;
   }
}

class Table extends Furniture{
   String type ;

   Table(String manufacturer, float price, String type){
      super(manufacturer,price);
      this.type = type;
   }
}
```

OUTPUT



10. Design classes having attributes and method(only skeleton) for a coffee shop. There are three different actors in our scenario i.e * Customer, Cashier,Barista.

```
package aayushi;
import java.util.ArrayList;
import java.util.List;
import java.util.PriorityQueue;
import java.util.Queue;

public class Question10 {
}


class Barista implements QueueOfPendingOrder,QueueOfCompletedOrder {
  String name;
  QueueOfPendingOrder queueOfPendingOrder;
  QueueOfCompletedOrder queueOfCompletedOrder;
  @Override
  public void remove(Order order) {
     queueOfPendingOrder.remove(order);
     System.out.println(order + "removed from Pending queue");
  }
  public void makeCoffee(Order order){
     System.out.println("Making coffee for " + order);
  }
  public void notifyAboutCompletedOrder(Customer customer,Order order) {
     System.out.println(customer + "your order " + order + " has been completed");
  }
  @Override
  public void addToCompleteOrderQueue(Order order) {
     queueOfCompletedOrder.addToCompleteOrderQueue(order);
     System.out.println(order + " added to Completed queue");
  }
}
class Cashier extends Order implements QueueOfPendingOrder {
  String name;
  QueueOfPendingOrder queueOfPendingOrder;
  List<Customer> customerList;
  public Cashier(String name,Long id) {
     super(id);
     this.name = name;
     customerList = new ArrayList<>();
  }
```

```
   String AcceptOrderAndAddCustomerToCustomerList(Customer customer,Order order,double
cash) {
      customerList.add(customer);
      System.out.println("Accepted order");
      return "token";
   }
   void addOrderInOrderQueue(Order order){
      queueOfPendingOrder.add(order);
      System.out.println(order + " added to order queue");
   }
}
class Customer {
   private String name;
   private String token;
   Cashier cashier;
   Order order;
   double amount;
   void placeOrder() {
      token = cashier.AcceptOrderAndAddCustomerToCustomerList(this,order,amount);
      System.out.println("This is the order token: " + token);
   }
   boolean waitingState(){
      System.out.println("Customer" + this.name + "is waiting");
      return true;
   }
   boolean drinkingState() {
      System.out.println("Customer " + this.name + " has collected coffee");
      return true;
   }
}
class Order {
   private Long id;
   public Order(Long id) {
      this.id = id;
   }
}
interface QueueOfPendingOrder {
   Queue<Order> queue = new PriorityQueue<>();
   default void add(Order order){
      queue.add(order);
   }
   default void remove(Order order) {
      queue.remove(order);
```

```
  }
}
interface QueueOfCompletedOrder {
  default void addToCompleteOrderQueue(Order order) {
     Queue<Order> queue = new PriorityQueue<>();
  }
}
```

11. Convert the following code so that it uses nested while statements instead of for statements.

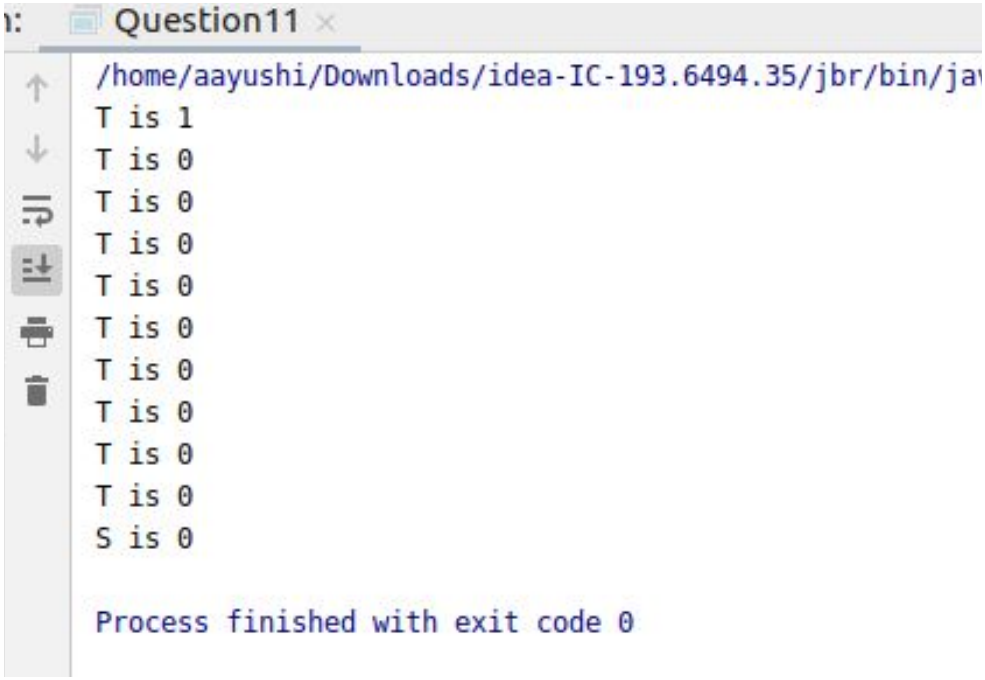
```
int s = 0;
   int t = 1;
   for (int i = 0; i < 10; i++)
   {
   s = s + i;
   for (int j = i; j > 0; j--)
   {
   t = t * (j - i);
   }
   s = s * t;
   System.out.println("T is " + t);
   }
   System.out.println("S is " + s);
```

CODE

```
package aayushi;
public class Question11 {
  public static void main(String[] args) {
     int s = 0;
     int t = 1;
     int i=0;
     int j;
     while(i < 10)
     {
        s = s + i;
        j = i;
        while(j > 0)
        {
           t = t * (j - i);
           j--;
        }
        s = s * t;
        System.out.println("T is " + t);
```

```
        i++;
      }
      System.out.println("S is " + s);
  }
}
```

OUTPUT



12.What will be the output on new Child(); ?

CODE

```
package aayushi;

class Parent extends Grandparent {
  {
     System.out.println("instance - parent");
  }
  public Parent() {
     System.out.println("constructor - parent");
  }
  static {
     System.out.println("static - parent");
  }
}
```

```
class Grandparent {
  static {
     System.out.println("static - grandparent");
  }
  {
     System.out.println("instance - grandparent");
  }

  public Grandparent() {
     System.out.println("constructor - grandparent");
  }
}

class Child extends Parent {
  public Child() {
     System.out.println("constructor - child");
  }
  static {
     System.out.println("static - child");
  }
  {
     System.out.println("instance - child");
  }
}
public class Question12 {
  public static void main(String[] args) {
     new Child();
  }
}
```

OUTPUT

13. Create a custom exception that do not have any stack trace.

CODE

```
package aayushi;
import java.util.Scanner;
//Create a custom exception that do not have any stack trace.

public class Question13A {
   public static void main(String[] args) throws HandMadeException {
      Scanner sc = new Scanner(System.in);

      try {
         System.out.println("Press Enter to throw Exception");
         sc.nextLine();
         throw new HandMadeException("This is a Custom Exception");
      }
      catch (HandMadeException hme){
         System.out.println(hme.getMessage());
         System.out.println(hme.getStackTrace());
      }

   }
}

class HandMadeException extends Exception{
   public HandMadeException(String message){
      super(message);
   }
}
```

OUTPUT