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.

```
package Java_Assignment2;

    // Interface for Customer is made

public interface Library_customer {

  public void addCustomer(String customerName,String customerAddress,Books books);        // it will add customer to the library

  public void Display(); //Display the Customers details

  public int ReturnTime(int date_of_issue,int date_of_return);               //it will return duration for which customer can keep the book

}
```

```
package Java_Assignment2;

// Abstract Class for Performing some Actions on the Books

abstract public class Books_Operation {

  public abstract void add(String Book_name, String Author_name, String Publisher);   //for Adding new Book to the library


  public abstract void deleteBook(String Book_name);                              // Delete particular Book from Library


  public abstract void Display();                                                 // Display the Detail of the Book

}
```

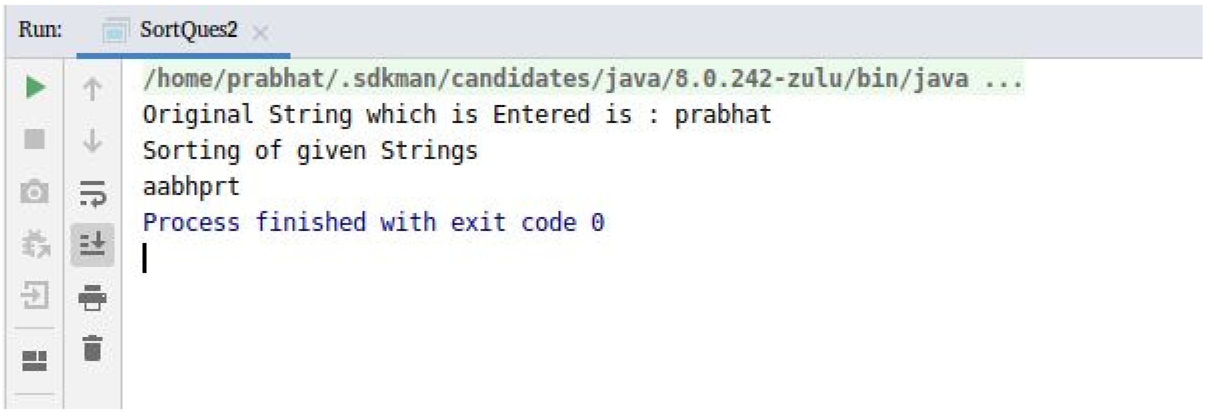
```
package Java_Assignment2;

public class Books extends Books_Operation {

  @Override

  public void add(String Book_name, String Author_name, String Publisher) {

  }
```

```java
  @Override
  public void deleteBook(String Book_name) {
  }
  @Override
  public void Display() {
  }
}
```

```java
package Java_Assignment2;
public class Customers implements Library_customer {
  @Override
  public void addCustomer(String customerName, String customerAddress, Books books) {
  }
  @Override
  public void Display() {
  }
  @Override
  public int ReturnTime(int date_of_issue, int date_of_return) {
      return 0;
  }
}
```

```java
package Java_Assignment2;
public class Library {
  public static void main(String[] args) {
        } }
```

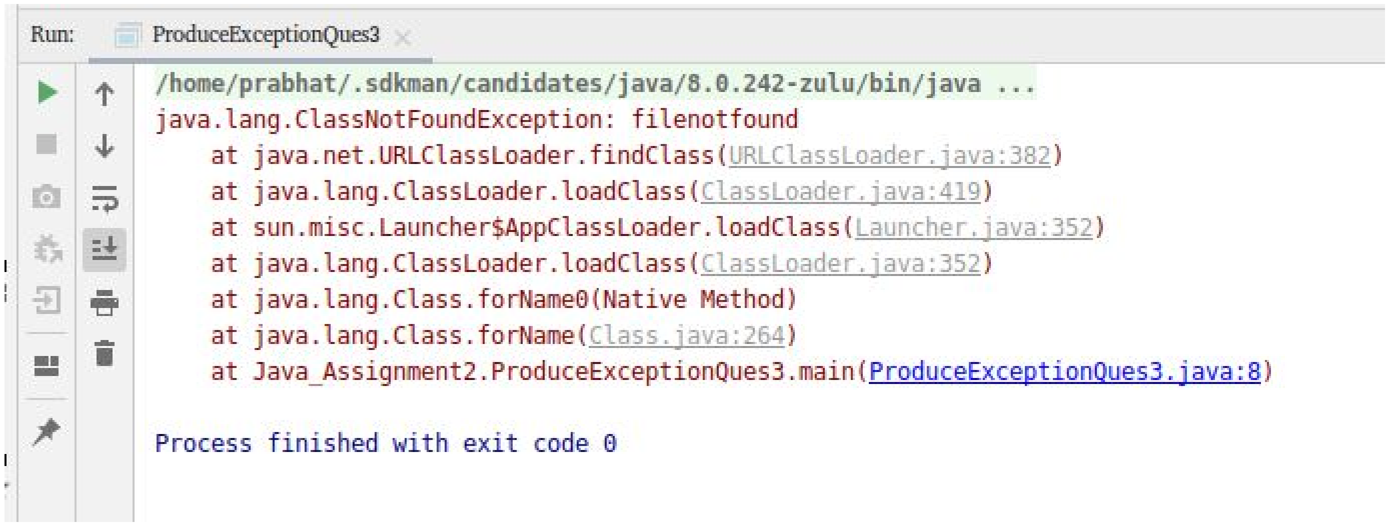
*2. WAP to sorting string without using string Methods?. (SortQues2.java)



```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Original String which is Entered is : prabhat
Sorting of given Strings
aabhprt
Process finished with exit code 0
```

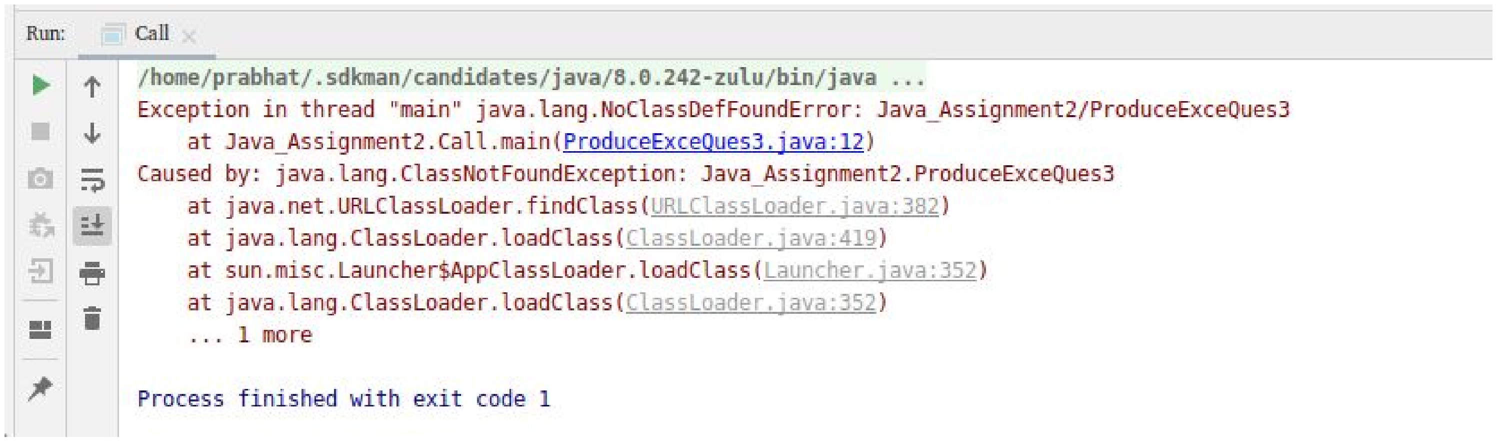
*3. WAP to produce NoClassDefFoundError and ClassNotFoundException exception.

For ClassNotFoundException (ProduceExceptionQues3.java)



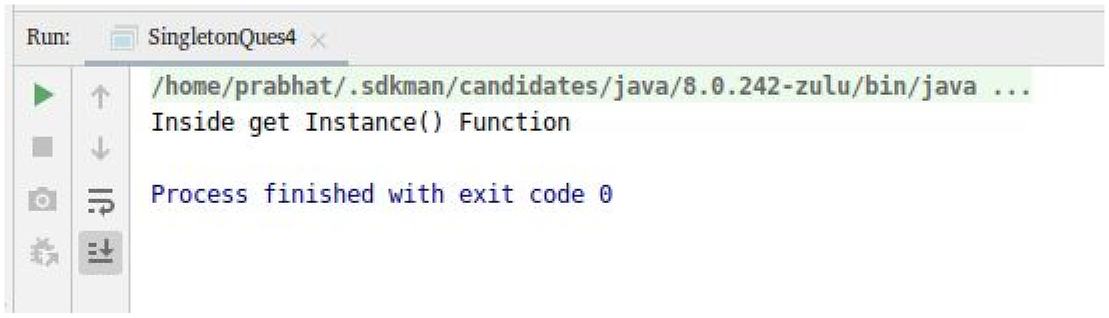
```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
java.lang.ClassNotFoundException: filenotfound
    at java.net.URLClassLoader.findClass(URLClassLoader.java:382)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:419)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:352)
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Class.java:264)
    at Java_Assignment2.ProduceExceptionQues3.main(ProduceExceptionQues3.java:8)

Process finished with exit code 0
```

For NoClassDeFoundError (ProduceExceQues3.java)



```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Exception in thread "main" java.lang.NoClassDefFoundError: Java_Assignment2/ProduceExceQues3
    at Java_Assignment2.Call.main(ProduceExceQues3.java:12)
Caused by: java.lang.ClassNotFoundException: Java_Assignment2.ProduceExceQues3
    at java.net.URLClassLoader.findClass(URLClassLoader.java:382)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:419)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:352)
    ... 1 more

Process finished with exit code 1
```

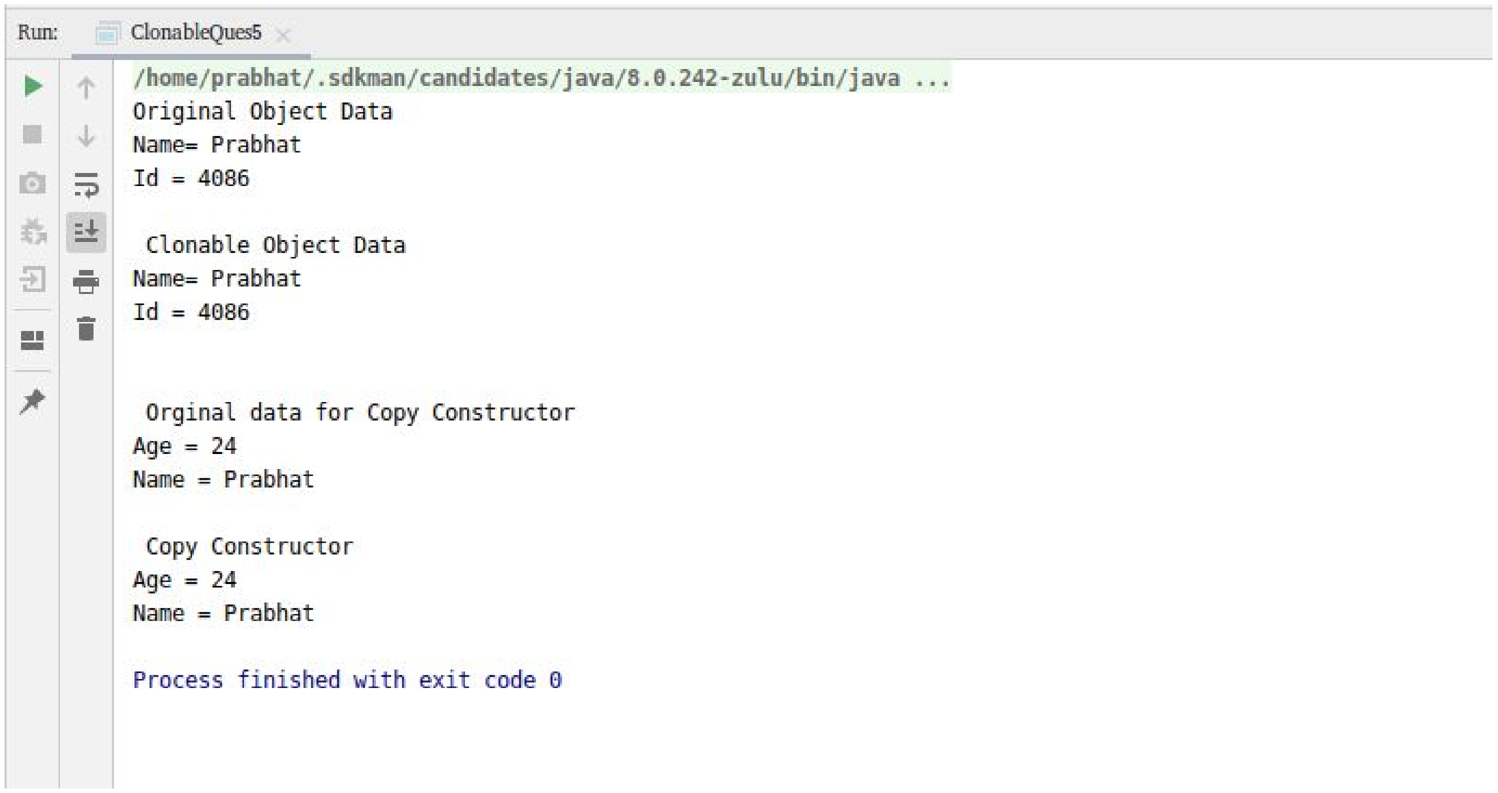
*4. WAP to create singleton class. (SingletonQues4.java)



```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Inside get Instance() Function

Process finished with exit code 0
```

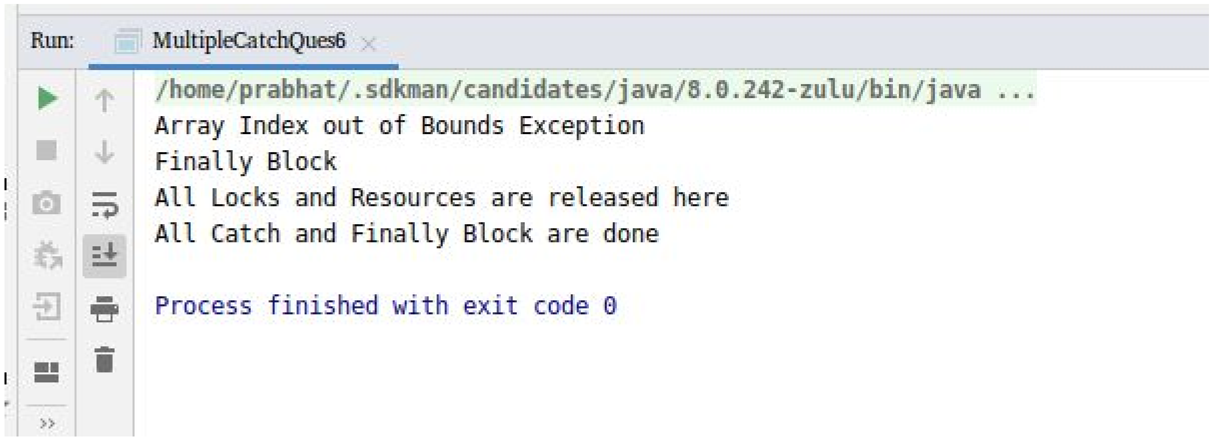
*5. WAP to show object cloning in java using cloneable and copy constructor both.

(Clonableques5.java)



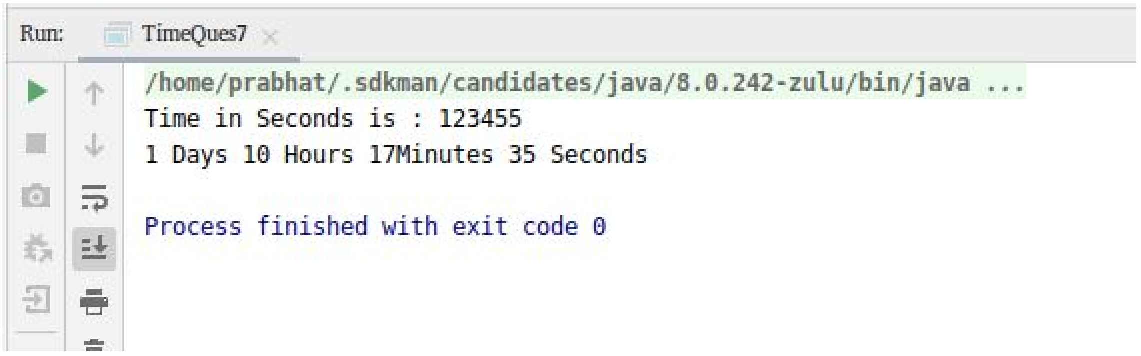
```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Original Object Data
Name= Prabhat
Id = 4086

 Clonable Object Data
Name= Prabhat
Id = 4086


 Orginal data for Copy Constructor
Age = 24
Name = Prabhat

 Copy Constructor
Age = 24
Name = Prabhat

Process finished with exit code 0
```

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



```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Array Index out of Bounds Exception
Finally Block
All Locks and Resources are released here
All Catch and Finally Block are done

Process finished with exit code 0
```

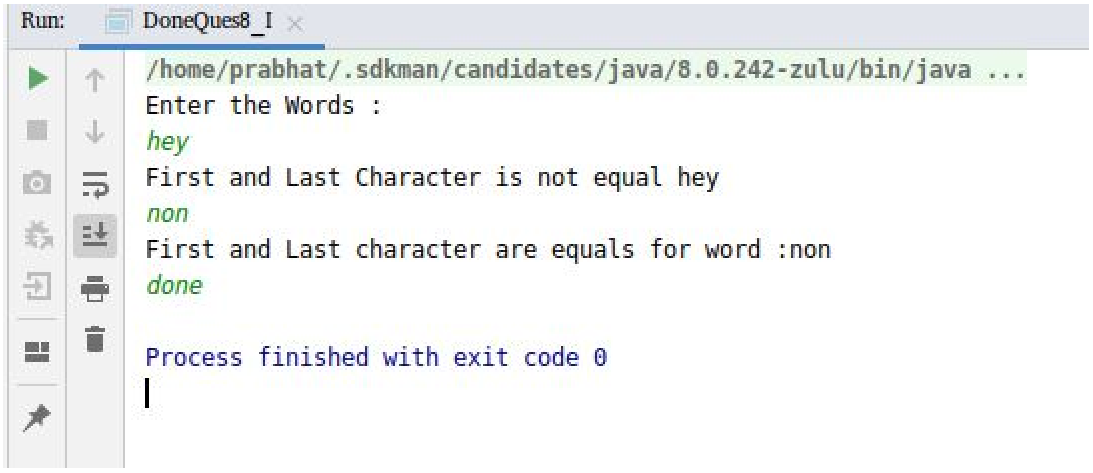
*7. WAP to convert seconds into days, hours, minutes and seconds.(TimeQues7.java)



```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Time in Seconds is : 123455
1 Days 10 Hours 17Minutes 35 Seconds

Process finished with exit code 0
```

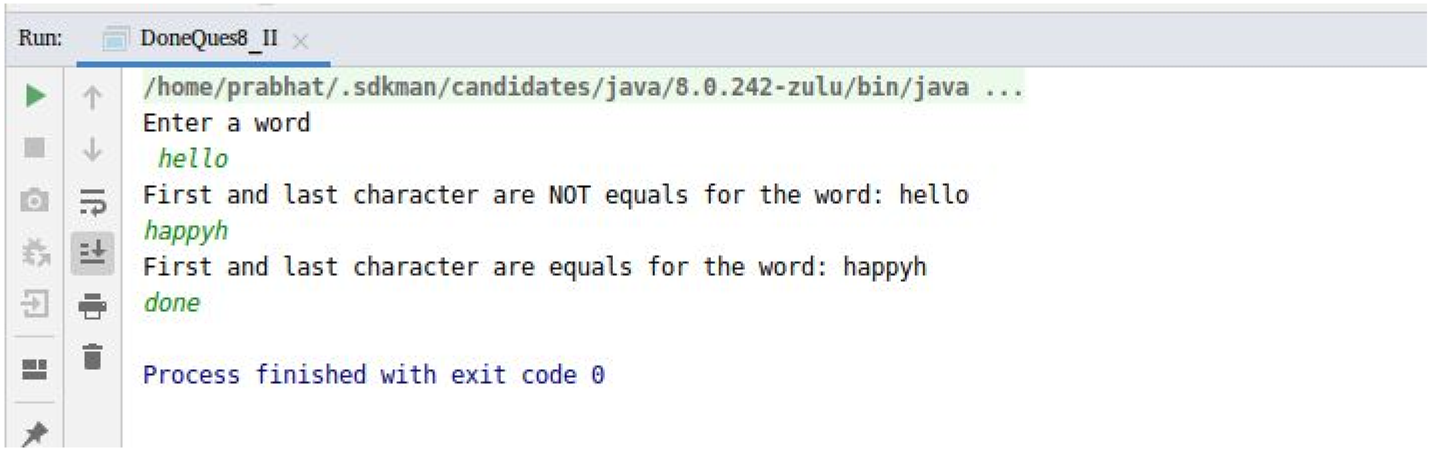
*8. WAP to read words from the keyboard until the word done is entered. For each word except done, report whether its first character is equal   to its last character. For the required loop, use a

a)while statement (DoneQues8_I.java)



```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Enter the Words :
hey
First and Last Character is not equal hey
non
First and Last character are equals for word :non
done

Process finished with exit code 0
```

b)do-while statement (DoneQues8_II.java)



```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Enter a word
 hello
First and last character are NOT equals for the word: hello
happyh
First and last character are equals for the word: happyh
done

Process finished with exit code 0
```

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.

```
package Java_Assignment2;

public interface Furniture {

  void StressTest();

  void FireTest();

}
```

```
package Java_Assignment2;

abstract public class Chair implements Furniture {

  public abstract String ChairType();

}
```

```
package Java_Assignment2;

abstract class Table implements Furniture {

  @Override

  public void StressTest() {
  }


  @Override
  public void FireTest() {


  }
  public abstract String Tabletype();
}
```

```
package Java_Assignment2;


public class WoodenChair extends Chair {
  @Override
  public String ChairType() {
    String string=" Wooden Chair";
    return string;
  }


  @Override
  public void StressTest() {
    System.out.println("Stress Test for WoodenChair is Approved");


  }


  @Override
  public void FireTest() {

     System.out.println("Fire Test for WoodenChair is Approved..");


  }

}
```

```
package Java_Assignment2;


public class WoodenTable extends Table {

  @Override

  public String Tabletype() {

     String string=" Wooden Table";

     return string;


  }


  @Override

  public void StressTest() {

     System.out.println("Stress Test for Wooden Table is Approved...");


  }


  @Override

  public void FireTest() {

     System.out.println("Fire Test for Wooden Table is Approved...");

  }
}
```

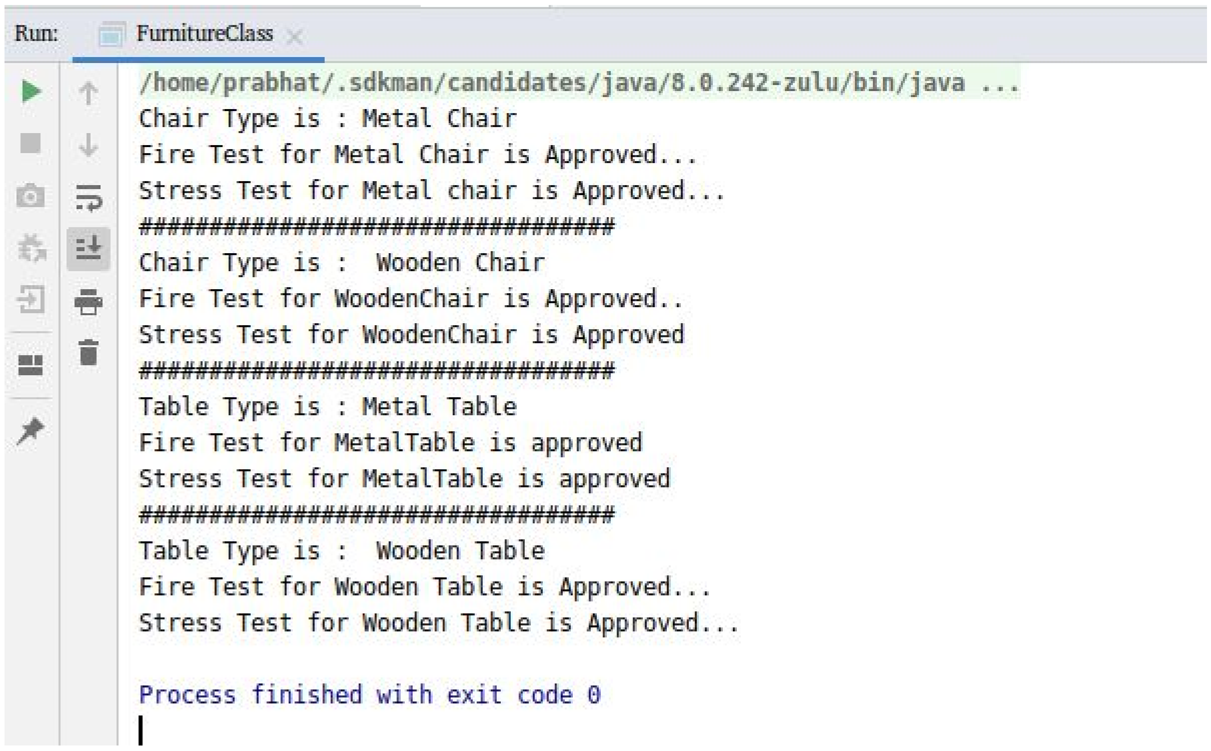
```
package Java_Assignment2;

import com.sun.security.auth.SolarisNumericUserPrincipal;

import jdk.nashorn.internal.parser.JSONParser;


public class MetalChair extends Chair {

  @Override

  public String ChairType() {

      String string="Metal Chair";

      return string;

  }


  @Override

  public void StressTest() {

      System.out.println("Stress Test for Metal chair is Approved...");


  }


  @Override

  public void FireTest() {

      System.out.println("Fire Test for Metal Chair is Approved...");

  }

}
```

```java
package Java_Assignment2;

public class MetalTable extends Table {
  @Override
  public String Tabletype() {
  String string="Metal Table";
  return string;
  }

  @Override
  public void StressTest() {
     System.out.println("Stress Test for MetalTable is approved ");

  }

  @Override
  public void FireTest() {
     System.out.println("Fire Test for MetalTable is approved ");

  }
}
```

```java
package Java_Assignment2;
```

```java
public class FurnitureClass {

  public static void main(String[] args) {

      MetalChair metalChair=new MetalChair();

      WoodenChair woodenChair=new WoodenChair();


      MetalTable metalTable=new MetalTable();

      WoodenTable woodenTable=new WoodenTable();




      System.out.println("Chair Type is : "+metalChair.ChairType());

      metalChair.FireTest();

      metalChair.StressTest();


      System.out.println("###################################");


      System.out.println("Chair Type is : "+woodenChair.ChairType());

      woodenChair.FireTest();

      woodenChair.StressTest();


      System.out.println("###################################");


      System.out.println("Table Type is : "+metalTable.Tabletype());

      metalTable.FireTest();

      metalTable.StressTest();
```

```
        System.out.println("##################################");



        System.out.println("Table Type is : "+woodenTable.Tabletype());

        woodenTable.FireTest();

        woodenTable.StressTest();



    }

}
```



```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Chair Type is : Metal Chair
Fire Test for Metal Chair is Approved...
Stress Test for Metal chair is Approved...
##################################
Chair Type is :  Wooden Chair
Fire Test for WoodenChair is Approved..
Stress Test for WoodenChair is Approved
##################################
Table Type is : Metal Table
Fire Test for MetalTable is approved
Stress Test for MetalTable is approved
##################################
Table Type is :  Wooden Table
Fire Test for Wooden Table is Approved...
Stress Test for Wooden Table is Approved...

Process finished with exit code 0
```

10. Design classes having attributes and method (only skeleton) for a coffee shop. There are three different actors in our scenario and i have listed the different actions they do also below

* Customer

  - Pays the cash to the cashier and places his order, get a token number back

  - Waits for the intimation that order for his token is ready

  - Upon intimation/notification he collects the coffee and enjoys his drink

  (Assumption: Customer waits till the coffee is done, he wont timeout and cancel the order. Customer always likes the drink served. Exceptions like he not liking his coffee, he getting wrong coffee are not considered to keep the design simple.)

* Cashier

  - Takes an order and payment from the customer

  - Upon payment, creates an order and places it into the order queue

  - Intimates the customer that he has to wait for his token and gives him his token

  (Assumption: Token returned to the customer is the order id. Order queue is unlimited. With a simple modification, we can design for a limited queue size)

* Barista

 - Gets the next order from the queue

 - Prepares the coffee

 - Places the coffee in the completed order queue

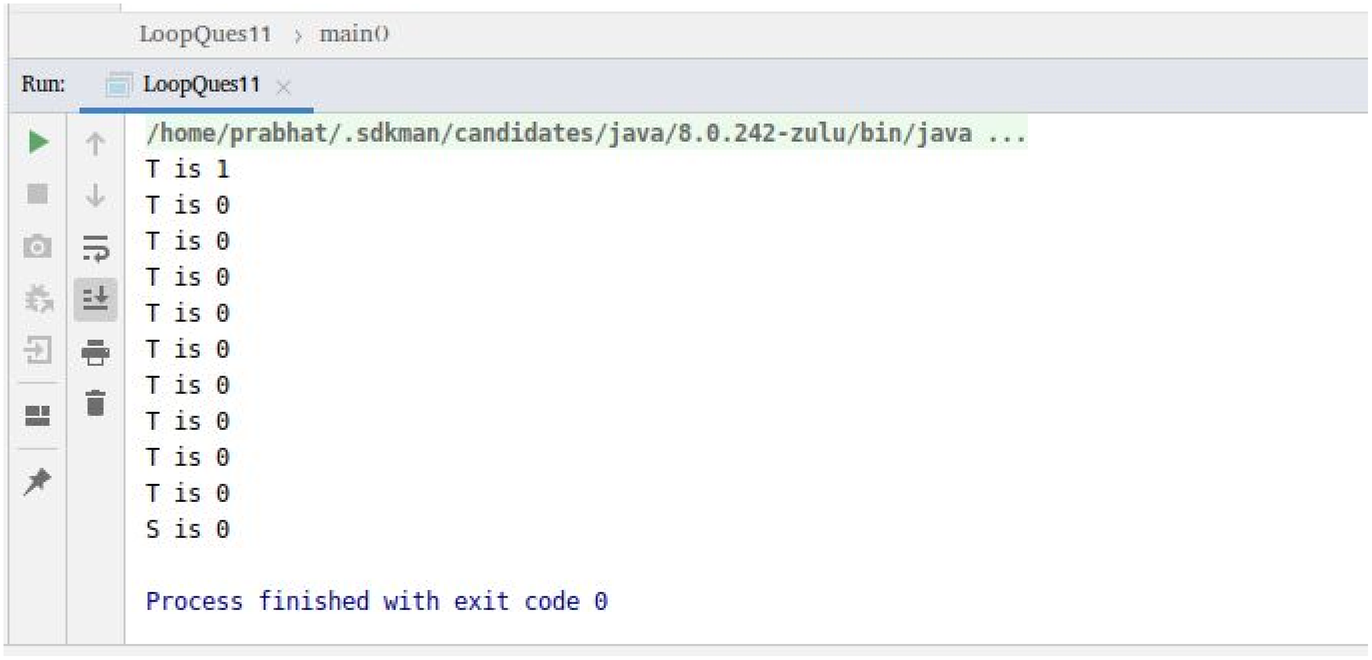
 - Places a notification that order for token is ready

```
package Java_Assignment2;

class Cafe {
  public int TokenNumber;

  public String order;

  private float payment;


  public int getTokenNumber()

  {

      return this.TokenNumber;

  }

  public void waitforCoffee(){}

  public void collectcoffee(){}

  public void paypayment(){}

  public void placeorder(){}

}


class Cashier{

  private String Order;

  private String getOrder()

  {

      return this.Order;

  }

  public void receivePayment(){}

  public void notify(Cafe customer){}


}

public class Customer{

  public static void main(String[] args) {
  }
}
```

*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);
```

(LoopQues11.java)



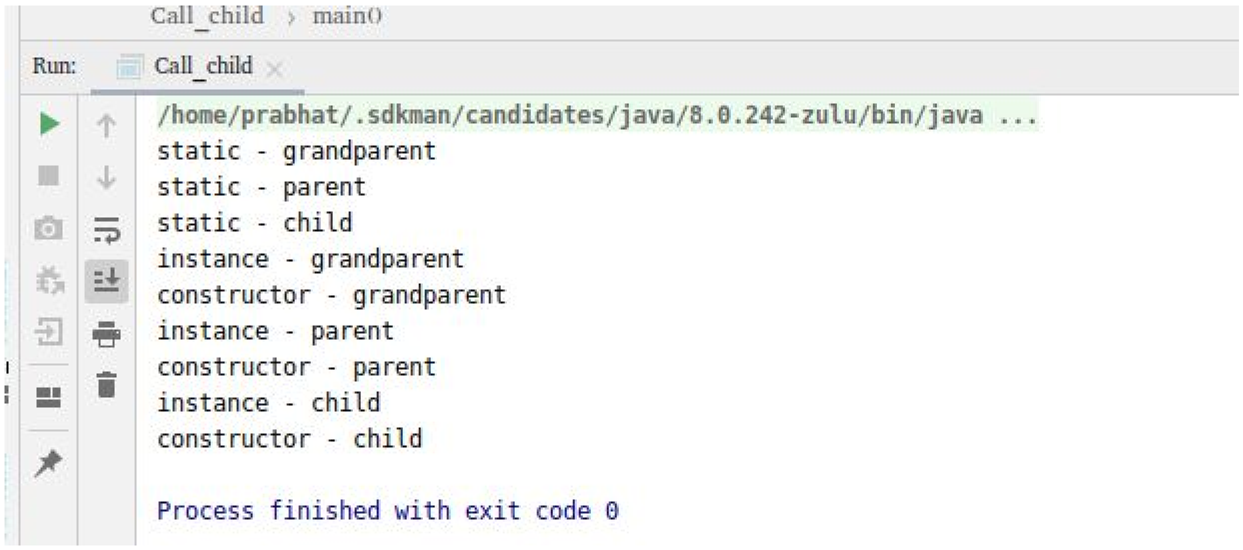
```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
T is 1
T is 0
T is 0
T is 0
T is 0
T is 0
T is 0
T is 0
T is 0
T is 0
S is 0

Process finished with exit code 0
```

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

```
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");
  }
```

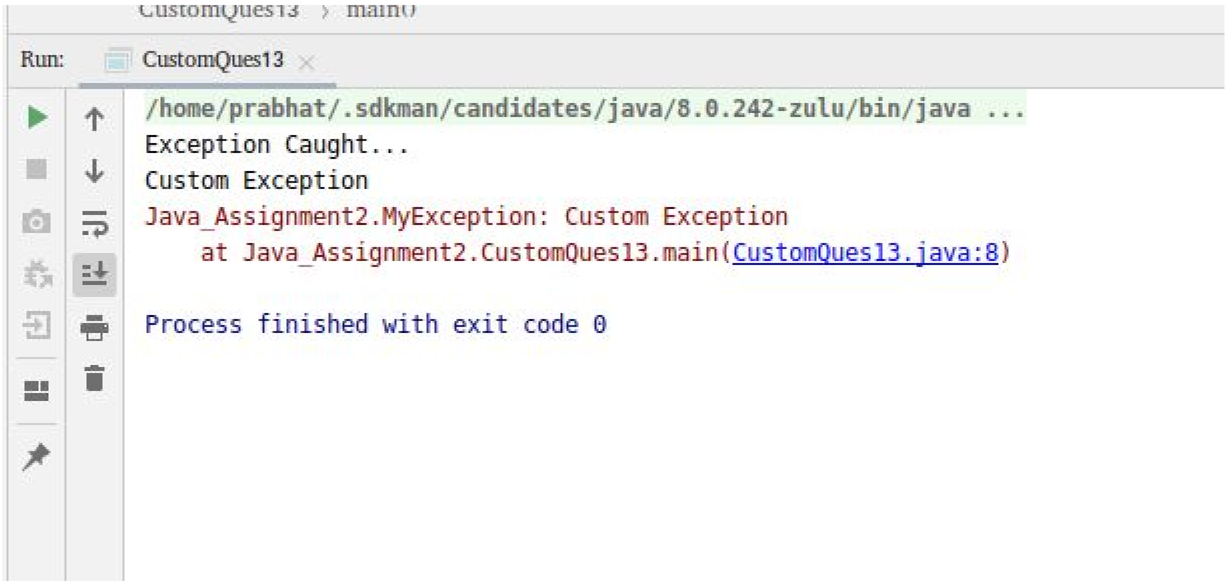
```
        {
          System.out.println("instance - child");
        }
    }
```

(GrandParentQues12.java)



```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
static - grandparent
static - parent
static - child
instance - grandparent
constructor - grandparent
instance - parent
constructor - parent
instance - child
constructor - child

Process finished with exit code 0
```

*Q13. Create a custom exception that do not have any stack trace.



```
/home/prabhat/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Exception Caught...
Custom Exception
Java_Assignment2.MyException: Custom Exception
	at Java_Assignment2.CustomQues13.main(CustomQues13.java:8)

Process finished with exit code 0
```