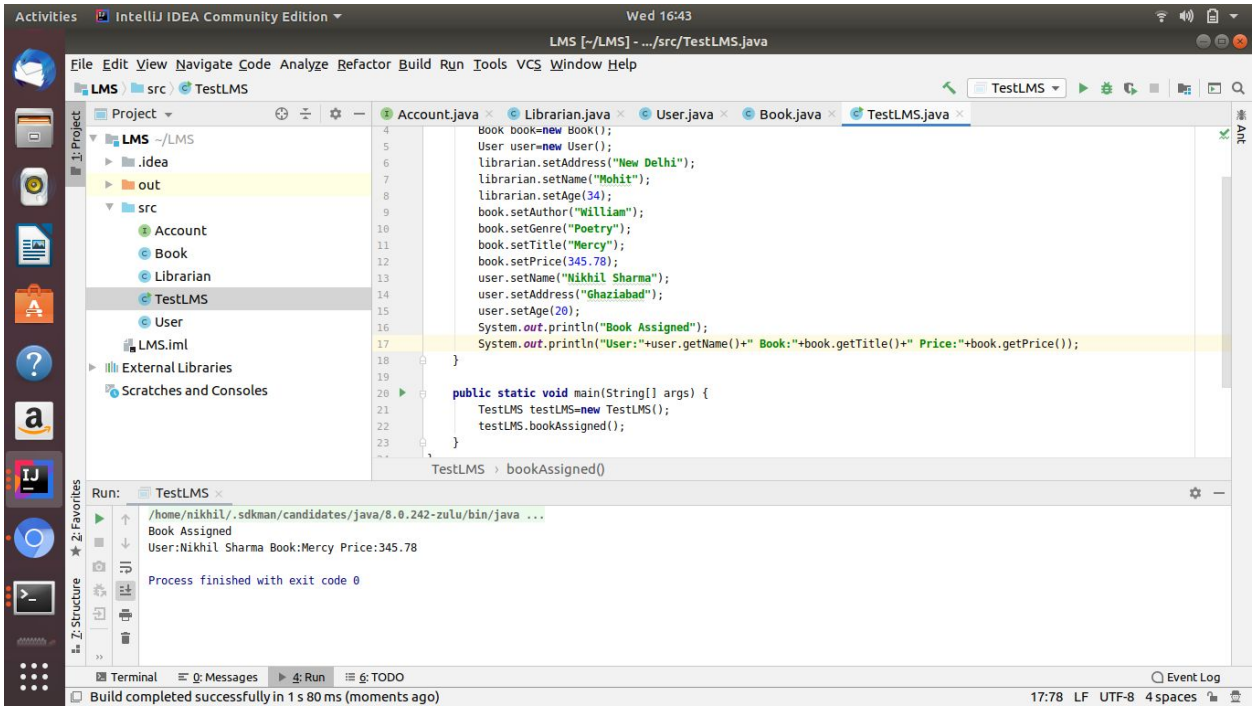
Q1. Create Java classes having suitable attributes for Library management system.Use OOPs concepts in your design.Also try to use interfaces and abstract classes.

Ans.



Code:-

Account.java:-

```
public interface Account {
  public void account();
}
```

Librarian.java:-

```
public class Librarian implements Account {
  String name;
  String Address;
  int age;
  @Override
  public void account() {
     String id;
     String pass;
  }

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }
```

```
  public String getAddress() {
     return Address;
  }

  public void setAddress(String address) {
     Address = address;
  }

  public int getAge() {
     return age;
  }

  public void setAge(int age) {
     this.age = age;
  }
}
```

User.java:-

```
public class User implements Account {
  int age;
  String name;
  String address;

  public int getAge() {
     return age;
  }

  public void setAge(int age) {
     this.age = age;
  }

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }

  public String getAddress() {
     return address;
  }

  public void setAddress(String address) {
     this.address = address;
  }

  @Override
  public void account() {
     String id;
```

```
        String pass;
    }

}
```

Book.java:-

```
public class Book {
    String author;
    String title;
    String genre;
    double price;

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getGenre() {
        return genre;
    }

    public void setGenre(String genre) {
        this.genre = genre;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
}
```

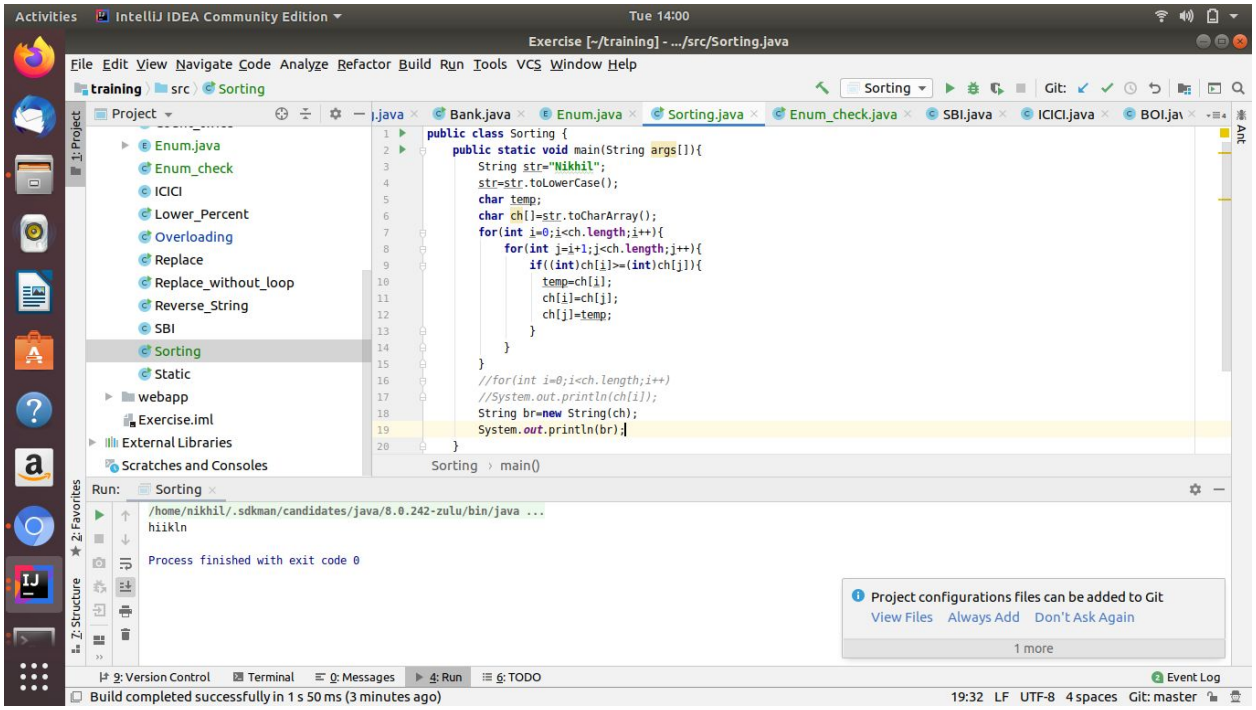
TestLMS.java:-

```
public class TestLMS {
    public void bookAssigned(){
        Librarian librarian=new Librarian();
        Book book=new Book();
```

```
        User user=new User();
        librarian.setAddress("New Delhi");
        librarian.setName("Mohit");
        librarian.setAge(34);
        book.setAuthor("William");
        book.setGenre("Poetry");
        book.setTitle("Mercy");
        book.setPrice(345.78);
        user.setName("Nikhil Sharma");
        user.setAddress("Ghaziabad");
        user.setAge(20);
        System.out.println("Book Assigned");
        System.out.println("User:"+user.getName()+" Book:"+book.getTitle()+" Price:"+book.getPrice());
    }

    public static void main(String[] args) {
        TestLMS testLMS=new TestLMS();
        testLMS.bookAssigned();
    }
}
```

Q2.WAP to sorting string without using string Methods?.

Ans.



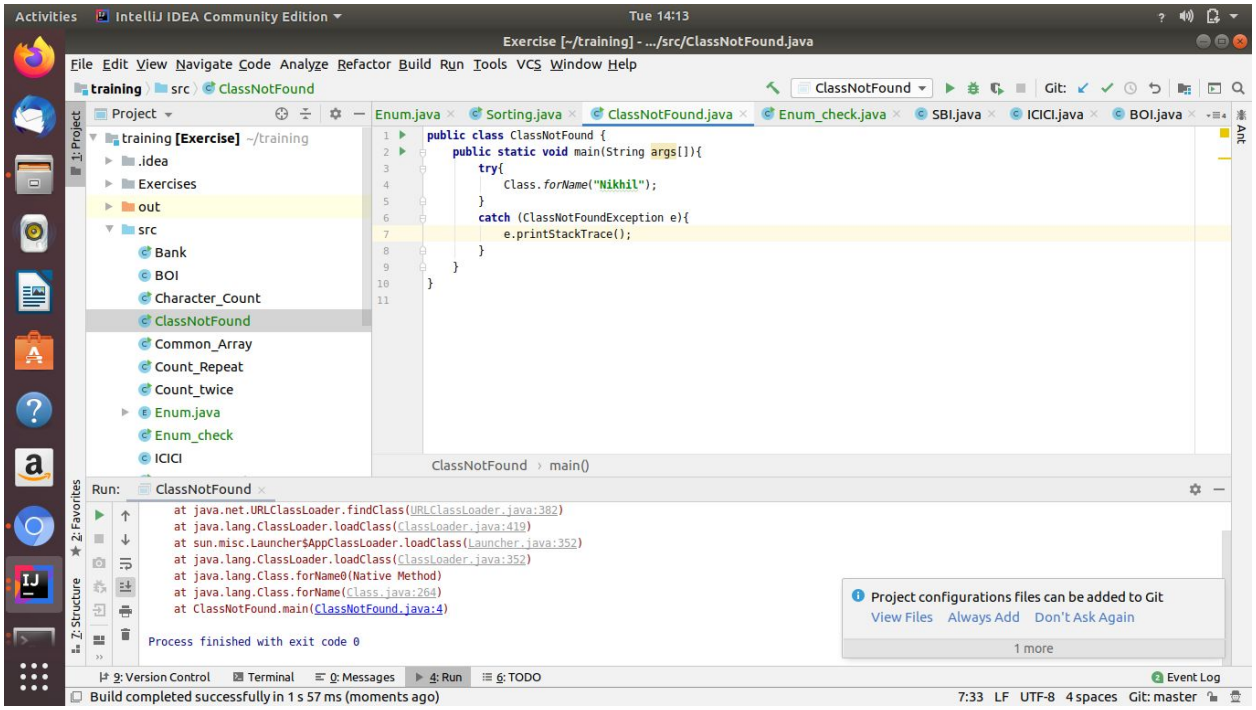
Code:-

```
public class Sorting {
    public static void main(String args[]){
        String str="Nikhil";
```

```
        str=str.toLowerCase();
        char temp;
        char ch[]=str.toCharArray();
        for(int i=0;i<ch.length;i++){
            for(int j=i+1;j<ch.length;j++){
                if((int)ch[i]>=(int)ch[j]){
                    temp=ch[i];
                    ch[i]=ch[j];
                    ch[j]=temp;
                }
            }
        }
        //for(int i=0;i<ch.length;i++)
        //System.out.println(ch[i]);
        String br=new String(ch);
        System.out.println(br);
    }
}
```

Q3.WAP to produce NoClassDefFoundError and ClassNotFoundException exception.

Ans.A) ClassNotFoundException

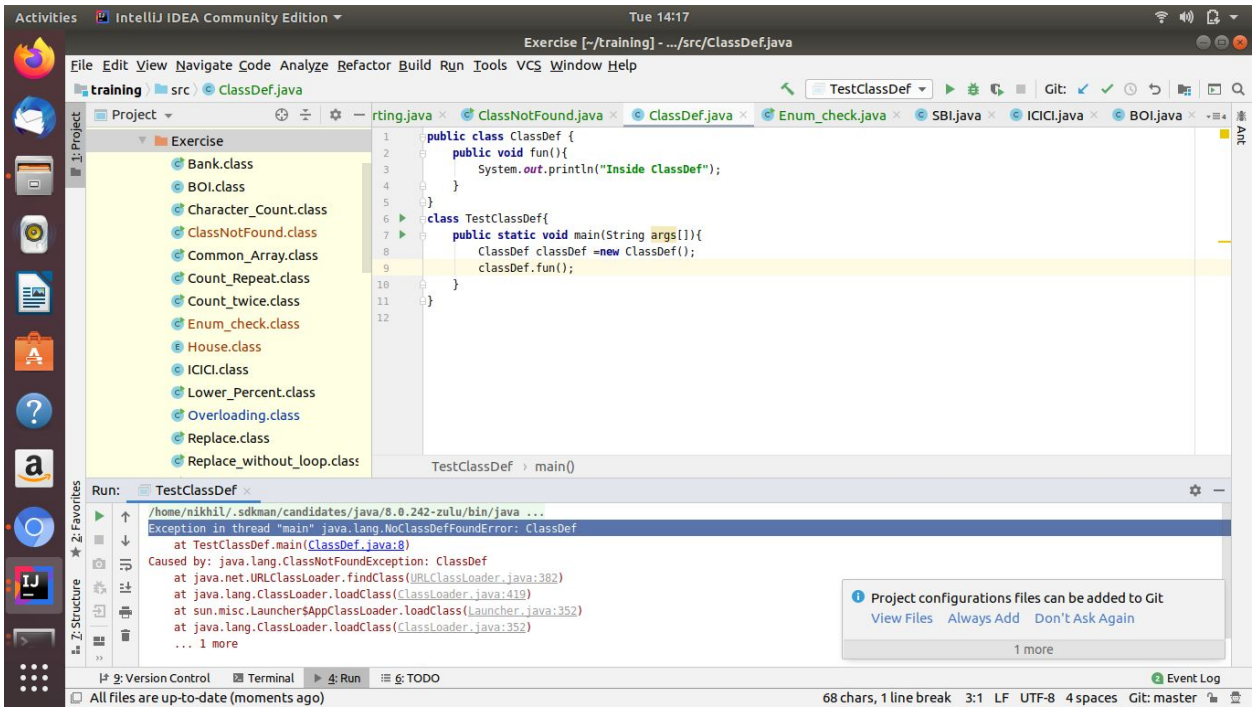


Code:-

```
public class ClassNotFound {
  public static void main(String args[]){
    try{
      Class.forName("Nikhil");
    }
    catch (ClassNotFoundException e){
      e.printStackTrace();
```

```
    }
  }
}
```

B)NoClassDefFound:-



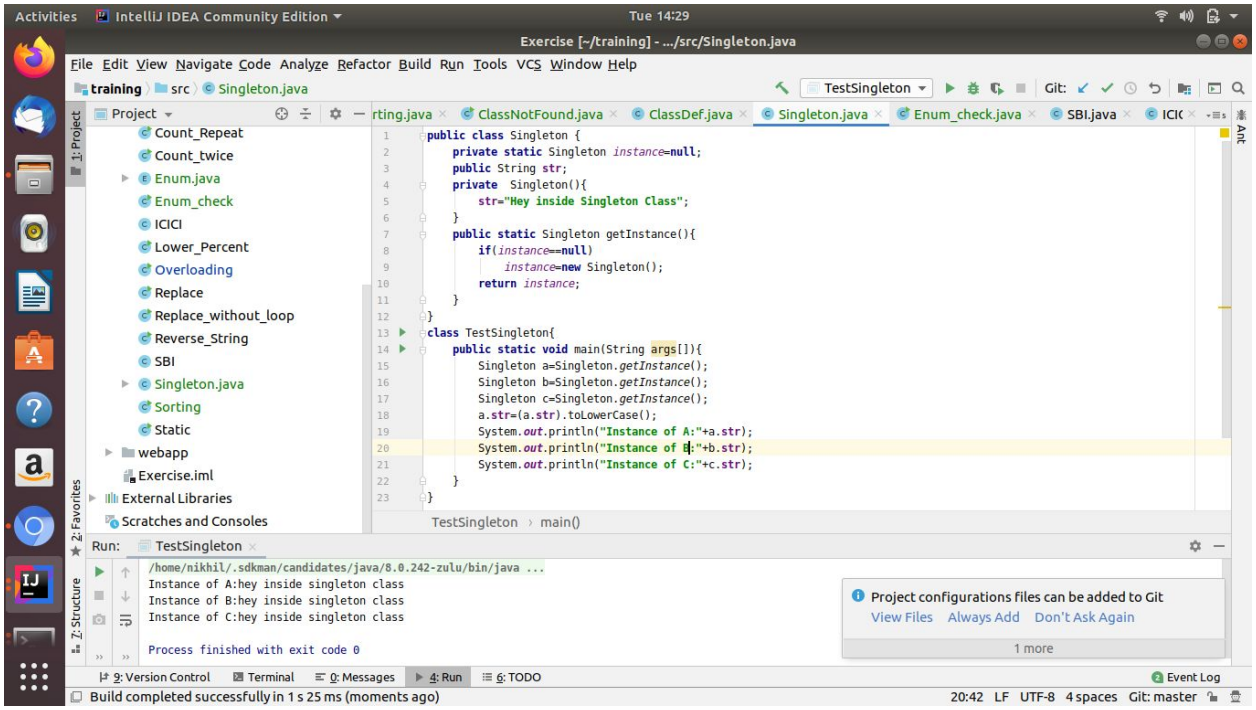
Code:-

```
public class ClassDef {
  public void fun(){
    System.out.println("Inside ClassDef");
  }
}
class TestClassDef{
  public static void main(String args[]){
    ClassDef classDef =new ClassDef();
    classDef.fun();
  }
}
```

Note:- After first compilation delete ClassDef.class file.

Q4.WAP to create singleton class.

Ans.



Code:-

Singleton.java:-

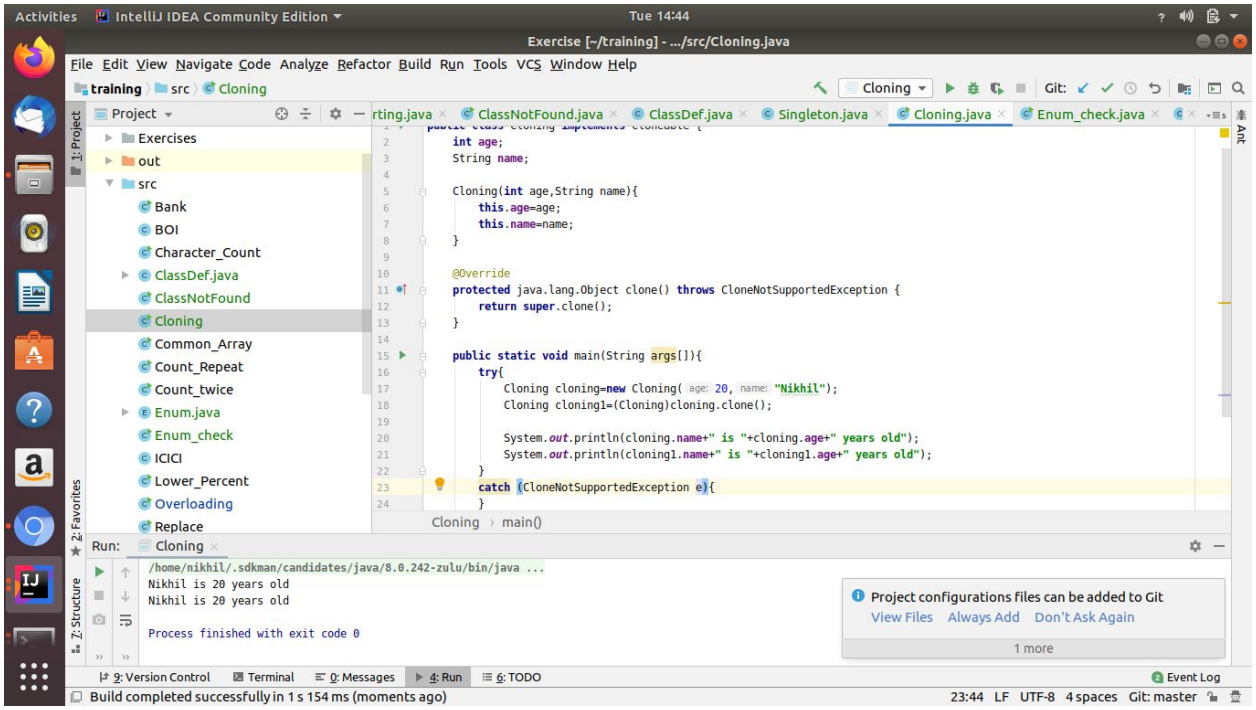
```
public class Singleton {
  private static Singleton instance=null;
  public String str;
  private  Singleton(){
     str="Hey inside Singleton Class";
  }
  public static Singleton getInstance(){
     if(instance==null)
        instance=new Singleton();
     return instance;
  }
}
```

TestSingleton.java:-

```
class TestSingleton{
  public static void main(String args[]){
     Singleton a=Singleton.getInstance();
     Singleton b=Singleton.getInstance();
     Singleton c=Singleton.getInstance();
     a.str=(a.str).toLowerCase();
     System.out.println("Instance of A:"+a.str);
     System.out.println("Instance of B:"+b.str);
     System.out.println("Instance of C:"+c.str);
  }
}
```

Q5.WAP to show object cloning in java using cloneable and copy constructor both.

Ans.

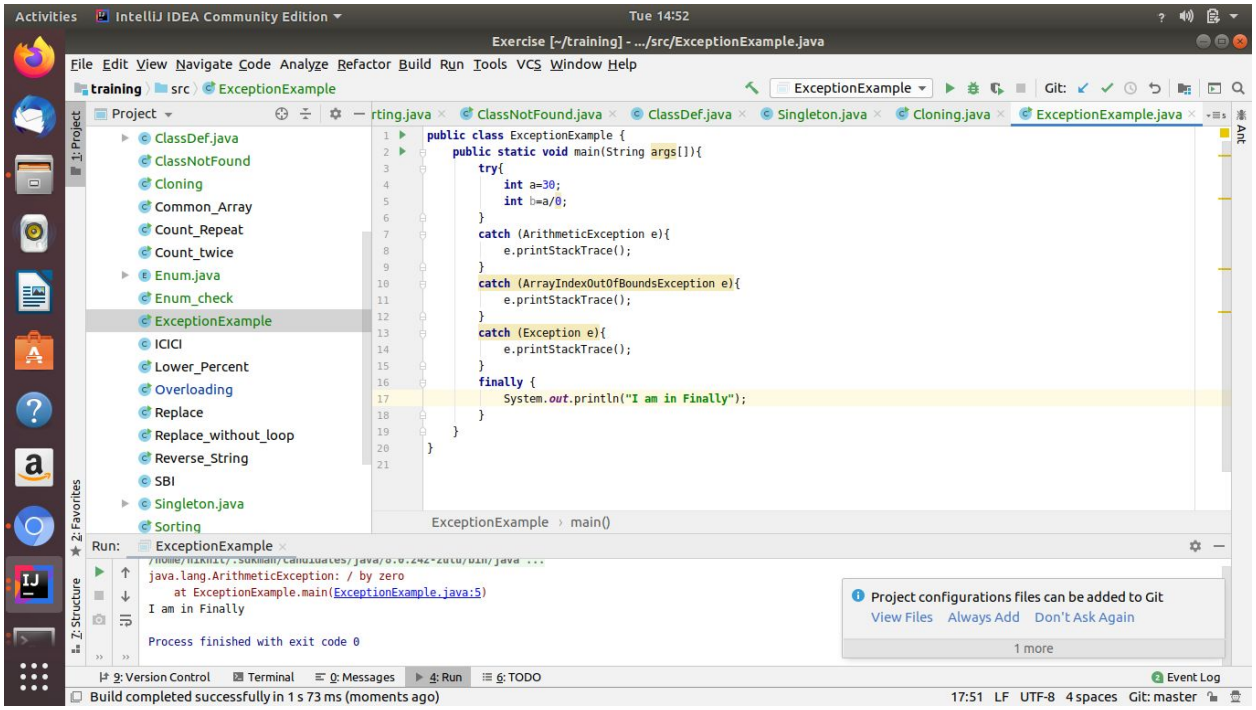


Code:-

```
public class Cloning implements Cloneable {
  int age;
  String name;

  Cloning(int age,String name){
     this.age=age;
     this.name=name;
  }

  @Override
  protected java.lang.Object clone() throws CloneNotSupportedException {
     return super.clone();
  }

  public static void main(String args[]){
     try{
        Cloning cloning=new Cloning(20,"Nikhil");
        Cloning cloning1=(Cloning)cloning.clone();

        System.out.println(cloning.name+" is "+cloning.age+" years old");
        System.out.println(cloning1.name+" is "+cloning1.age+" years old");
     }
     catch (CloneNotSupportedException e){
     }
  }
}
```

Q6.WAP showing try, multi-catch and finally blocks.

Ans.

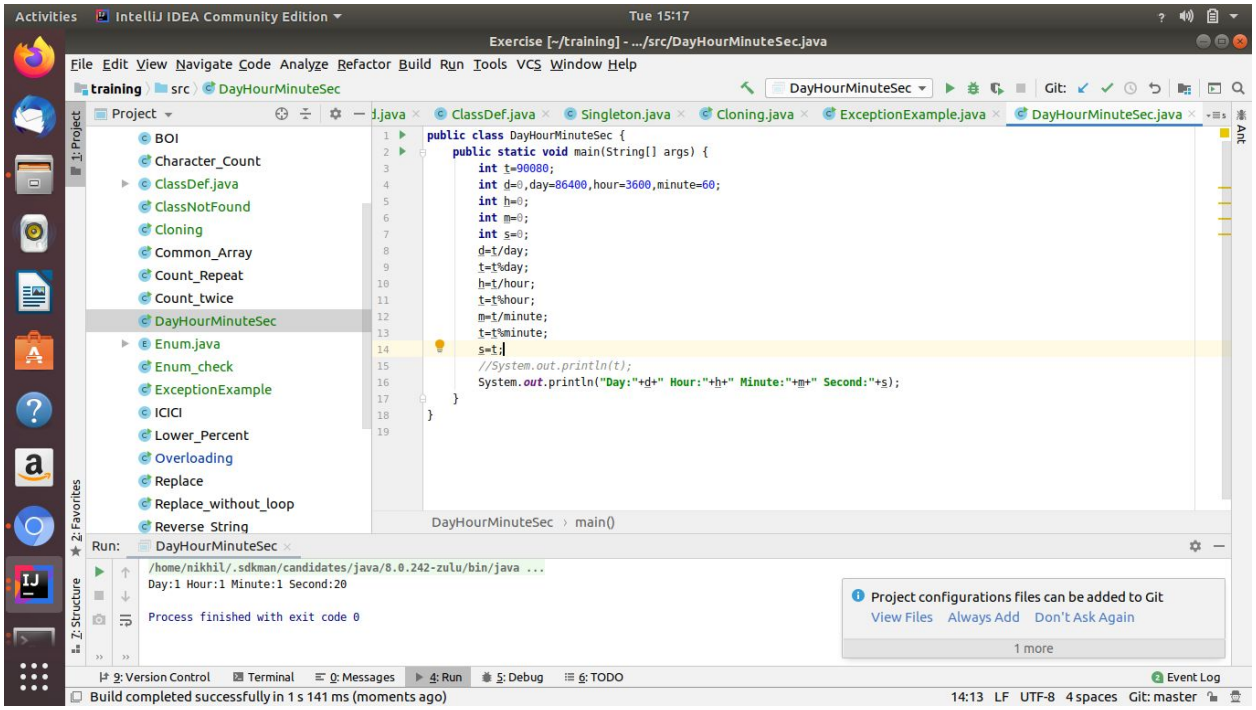


Code:-

```
public class ExceptionExample {
  public static void main(String args[]){
    try{
      int a=30;
      int b=a/0;
    }
    catch (ArithmeticException e){
      e.printStackTrace();
    }
    catch (ArrayIndexOutOfBoundsException e){
      e.printStackTrace();
    }
    catch (Exception e){
      e.printStackTrace();
    }
    finally {
      System.out.println("I am in Finally");
    }
  }
}
```

Q7.WAP to convert seconds into days, hours, minutes and seconds.

Ans.



Code:-

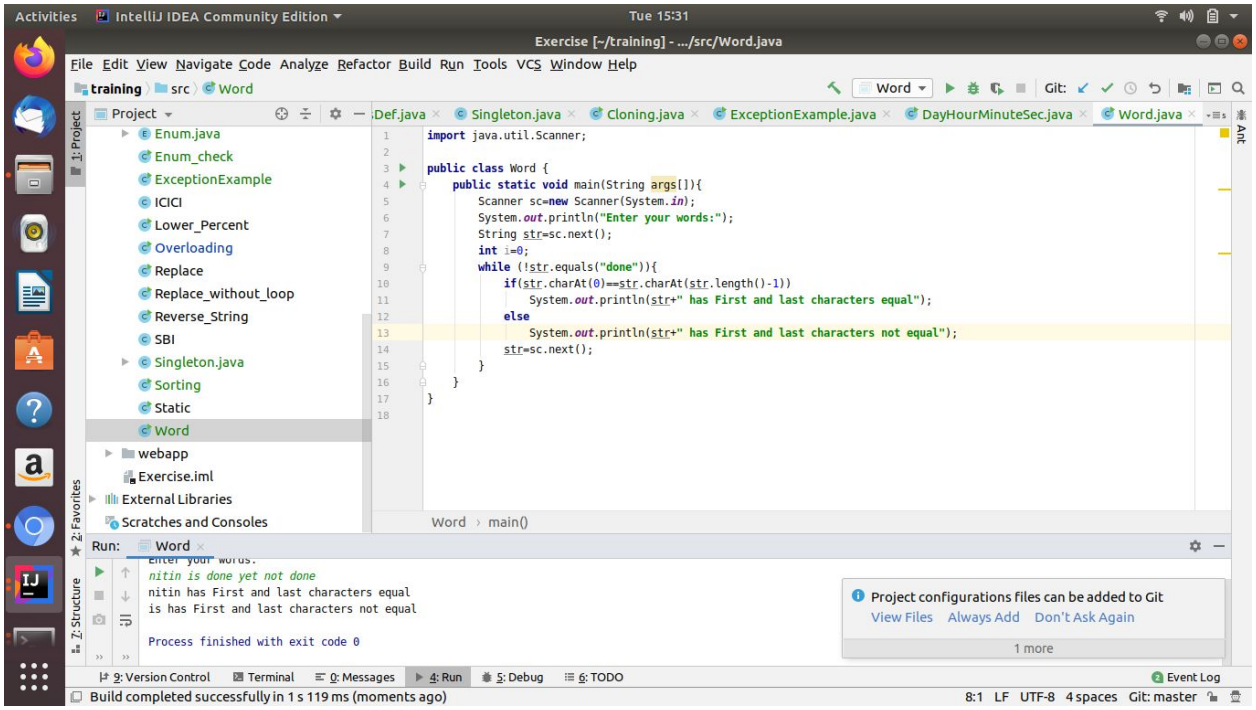
```
public class DayHourMinuteSec {
  public static void main(String[] args) {
     int t=90080;
     int d=0,day=86400,hour=3600,minute=60;
     int h=0;
     int m=0;
     int s=0;
     d=t/day;
     t=t%day;
     h=t/hour;
     t=t%hour;
     m=t/minute;
     t=t%minute;
     s=t;
     //System.out.println(t);
     System.out.println("Day:"+d+" Hour:"+h+" Minute:"+m+" Second:"+s);
  }
}
```

Q8.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

b)do-while statement

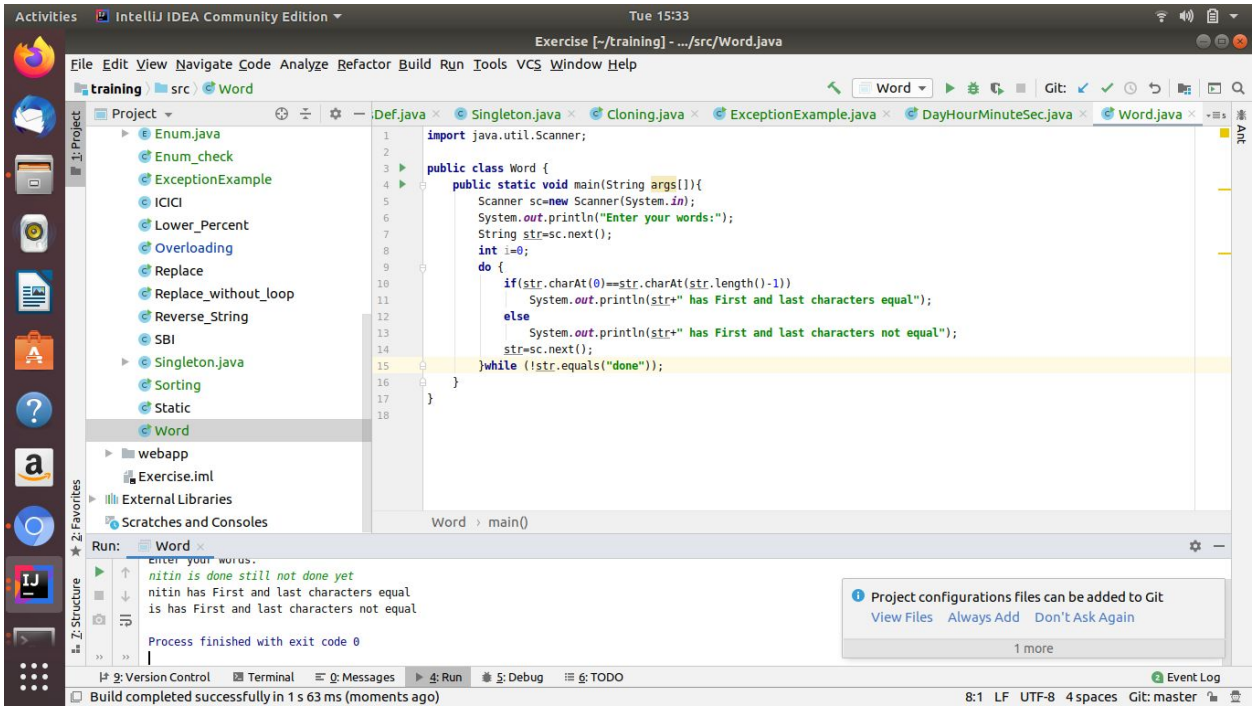
Ans.



Using While:-

```
import java.util.Scanner;

public class Word {
  public static void main(String args[]){
     Scanner sc=new Scanner(System.in);
     System.out.println("Enter your words:");
     String str=sc.next();
     int i=0;
     while (!str.equals("done")){
        if(str.charAt(0)==str.charAt(str.length()-1))
           System.out.println(str+" has First and last characters equal");
        else
           System.out.println(str+" has First and last characters not equal");
        str=sc.next();
     }
  }
}
```

Using Do while:-

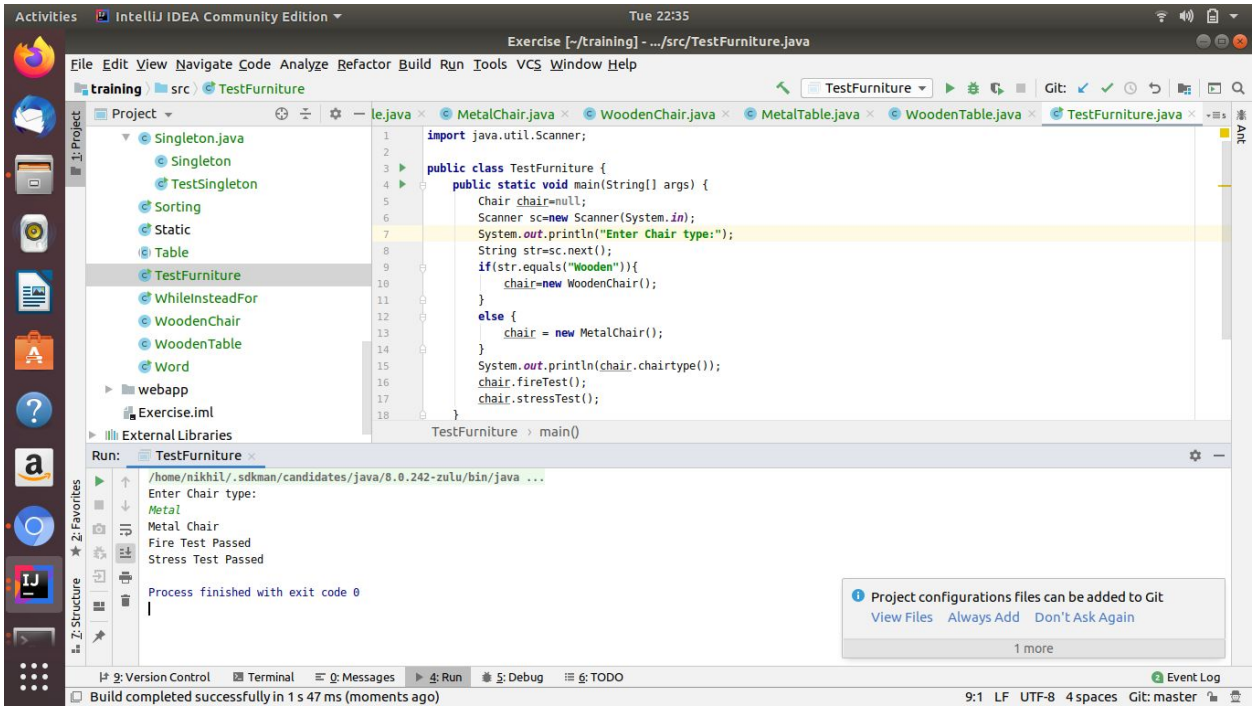


Code:-

```
import java.util.Scanner;

public class Word {
  public static void main(String args[]){
     Scanner sc=new Scanner(System.in);
     System.out.println("Enter your words:");
     String str=sc.next();
     int i=0;
     do {
        if(str.charAt(0)==str.charAt(str.length()-1))
           System.out.println(str+" has First and last characters equal");
        else
           System.out.println(str+" has First and last characters not equal");
        str=sc.next();
     }while (!str.equals("done"));
  }
}
```

Q9.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.

Ans.



Furniture.java:-

```
public interface Furniture {
  public void stressTest();
  public void fireTest();
}
```

Table.java:-

```
public abstract class Table implements Furniture {
  public abstract String tableType();
}
```

Chair.java:-

```
public abstract class Chair implements Furniture {
  public abstract String chairtype();
}
```

MetalTable.java:-

```
public class MetalTable extends Table {
  @Override
  public void stressTest() {
     System.out.println("Stress Test Passed....");
  }

  @Override
  public void fireTest() {
     System.out.println("Fire Test Passed....");
  }
```

```
  @Override
  public String tableType() {
     String s="Metal Table";
     return s;
  }
}
```

WoodenTable.java:-

```
public class WoodenTable extends Table {
  @Override
  public void stressTest() {
     System.out.println("Stress Test Passed....");
  }

  @Override
  public void fireTest() {
     System.out.println("Fire Test failed.... ");
  }

  @Override
  public String tableType() {
     String s="Wooden Table";
     return s;
  }
}
```

MetalChair.java:-

```
public class MetalChair extends Chair {
  public void stressTest(){
     System.out.println("Stress Test Passed");
  }
  public void fireTest(){
     System.out.println("Fire Test Passed");
  }

  @Override
  public String chairtype() {
     String s="Metal Chair";
     return s;
  }

}
```

WoodenChair.java:-

```
public class WoodenChair extends Chair {
  @Override
  public void stressTest() {
     System.out.println("Stress Test Failed");
  }

  @Override
  public void fireTest() {
```

```
        System.out.println("Fire test Failed");
    }

    @Override
    public String chairtype() {
        String s="Wooden Chair";
        return s;
    }
}
```

TestFurniture.java:-

```
import java.util.Scanner;

public class TestFurniture {
    public static void main(String[] args) {
        Chair chair=null;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Chair type:");
        String str=sc.next();
        if(str.equals("Wooden")){
            chair=new WoodenChair();
        }
        else {
            chair = new MetalChair();
        }
        System.out.println(chair.chairtype());
        chair.fireTest();
        chair.stressTest();
    }
}
```

Q10.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

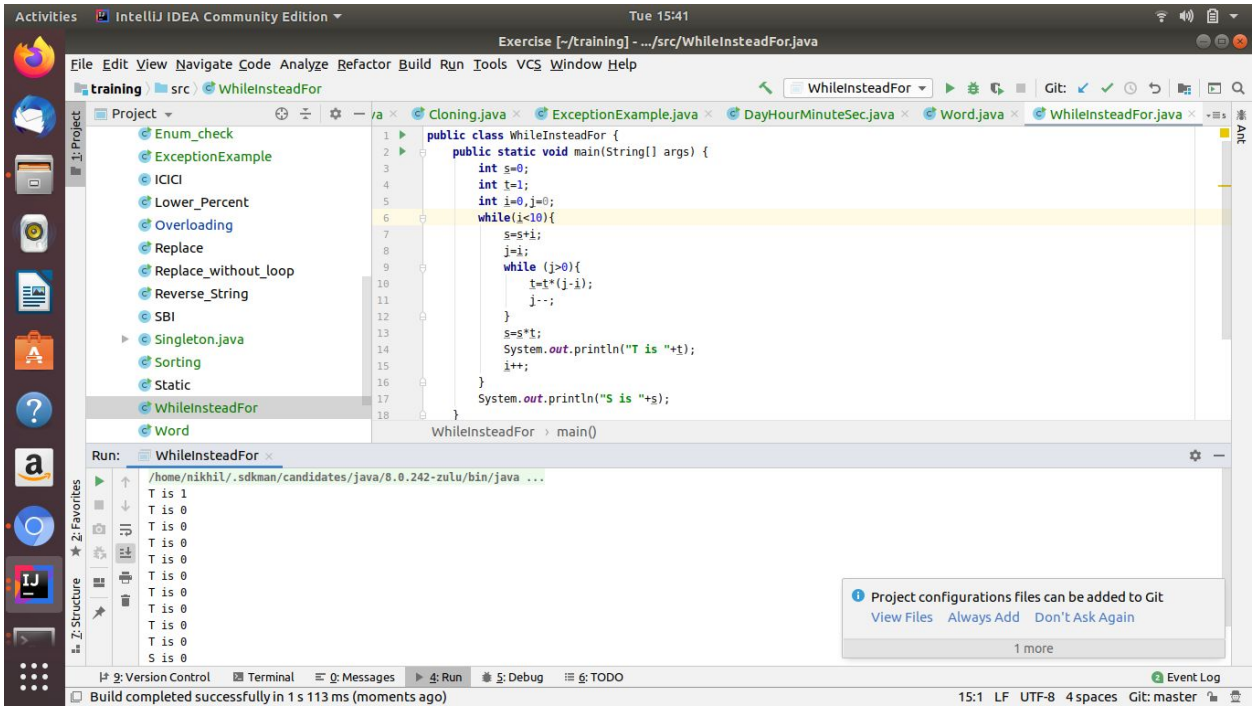
- Places a notification that order for token is ready

Ans.

Q11. 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);
```

Ans.



Code:-

```
public class WhileInsteadFor {
  public static void main(String[] args) {
     int s=0;
     int t=1;
     int i=0,j=0;
     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);
  }
}
```

Q12.What will be the output on new Child(); ?

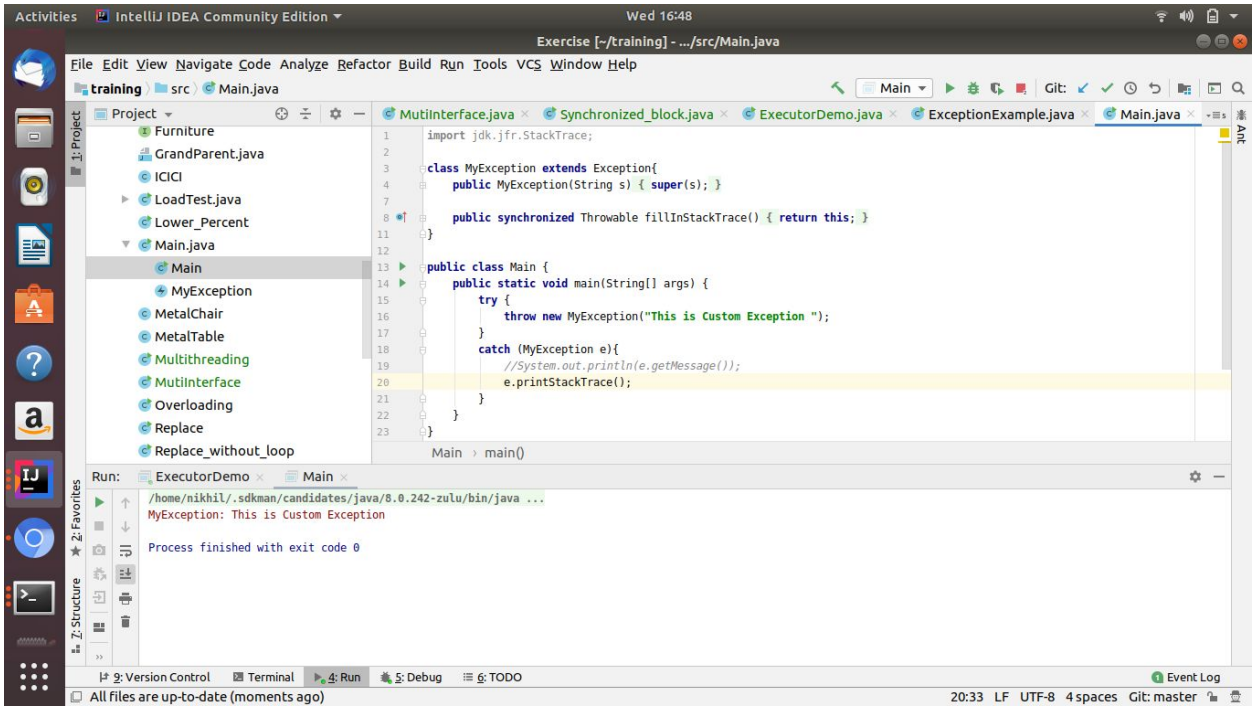
class Parent extends Grandparent {

```
    {
        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");
    }
  }
```

Ans.static-grandparent
Static-parent
Static-child
Instance-grandparent
Instance-parent
Instance-child
Constructor-grandparent
Constructor-parent
constructor-child

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



Code:-

```
import jdk.jfr.StackTrace;

class MyException extends Exception{
  public MyException(String s){
    super(s);
```

```
  }

  public synchronized Throwable fillInStackTrace(){
    return this;
  }
}

public class Main {
  public static void main(String[] args) {
    try {
      throw new MyException("This is Custom Exception ");
    }
    catch (MyException e){
      //System.out.println(e.getMessage());
      e.printStackTrace();
    }
  }
}
```