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.
2. WAP to sorting string without using string Methods?.
3. WAP to produce NoClassDefFoundError and ClassNotFoundException exception.
4. WAP to create singleton class.
5. WAP to show object cloning in java using cloneable and copy constructor both.
6. WAP showing try, multi-catch and finally blocks.
7. WAP to convert seconds into days, hours, minutes and seconds.
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
b)do-while statement

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.

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

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);
```

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");
    }
```

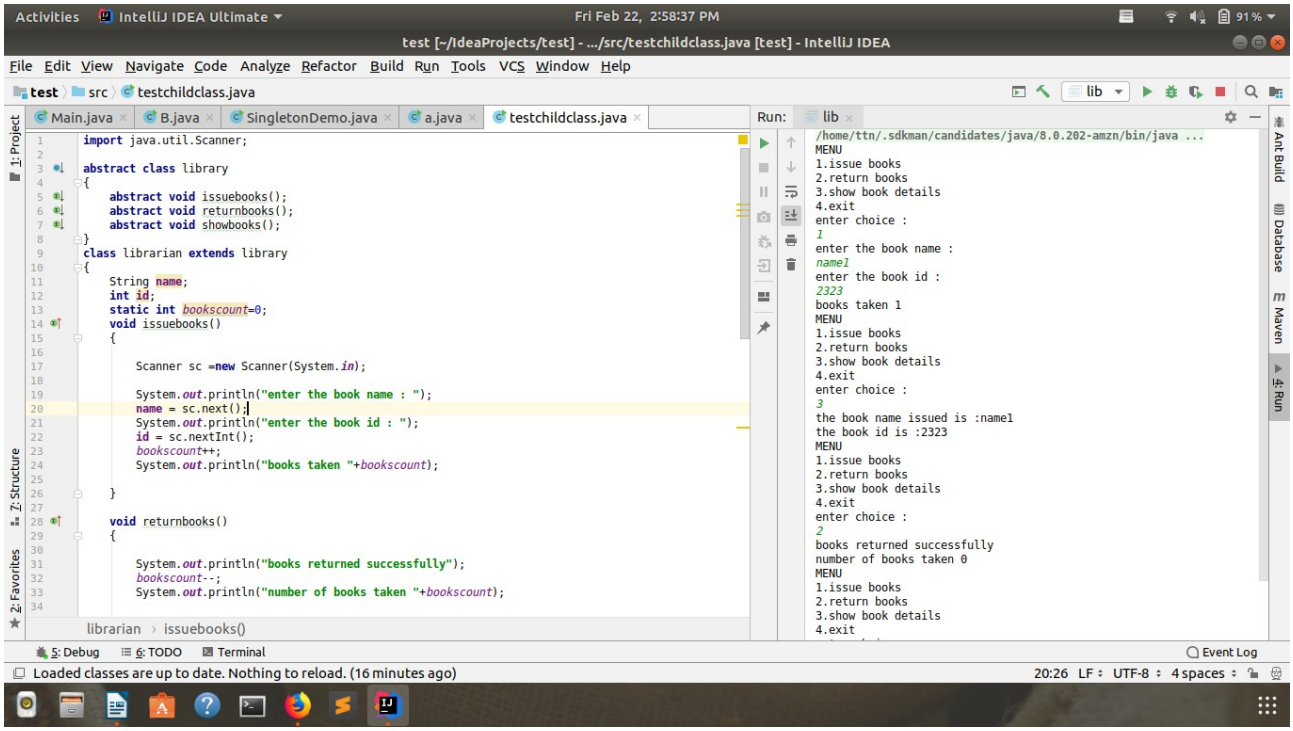
```
    {
    System.out.println("instance - child");
    }
  }
```

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

ANSWERS

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.

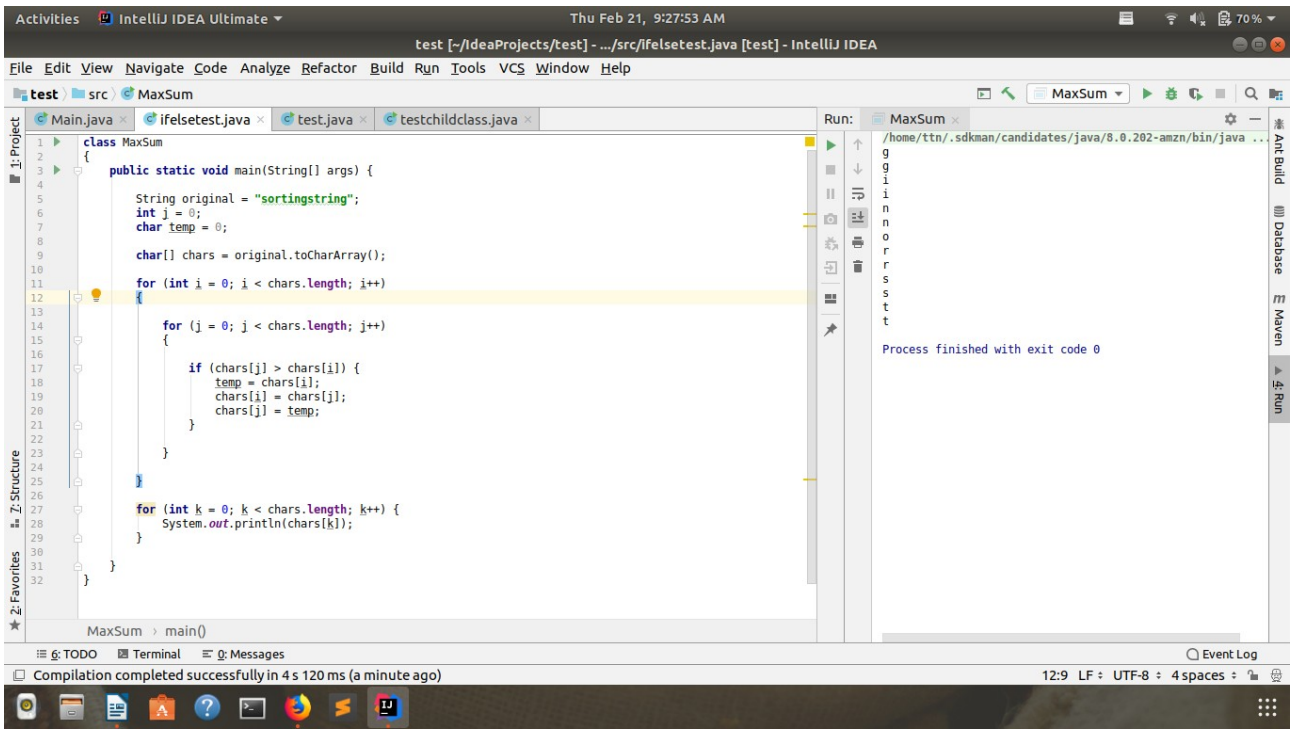
ANS.




```java
import java.util.Scanner;
abstract class library
{
    abstract void issuebooks();
    abstract void returnbooks();
    abstract void showbooks();
}
class librarian extends library
{
    String name;
    int id;
    static int bookscount=0;
    void issuebooks()
    {

        Scanner sc =new Scanner(System.in);

        System.out.println("enter the book name : ");
```

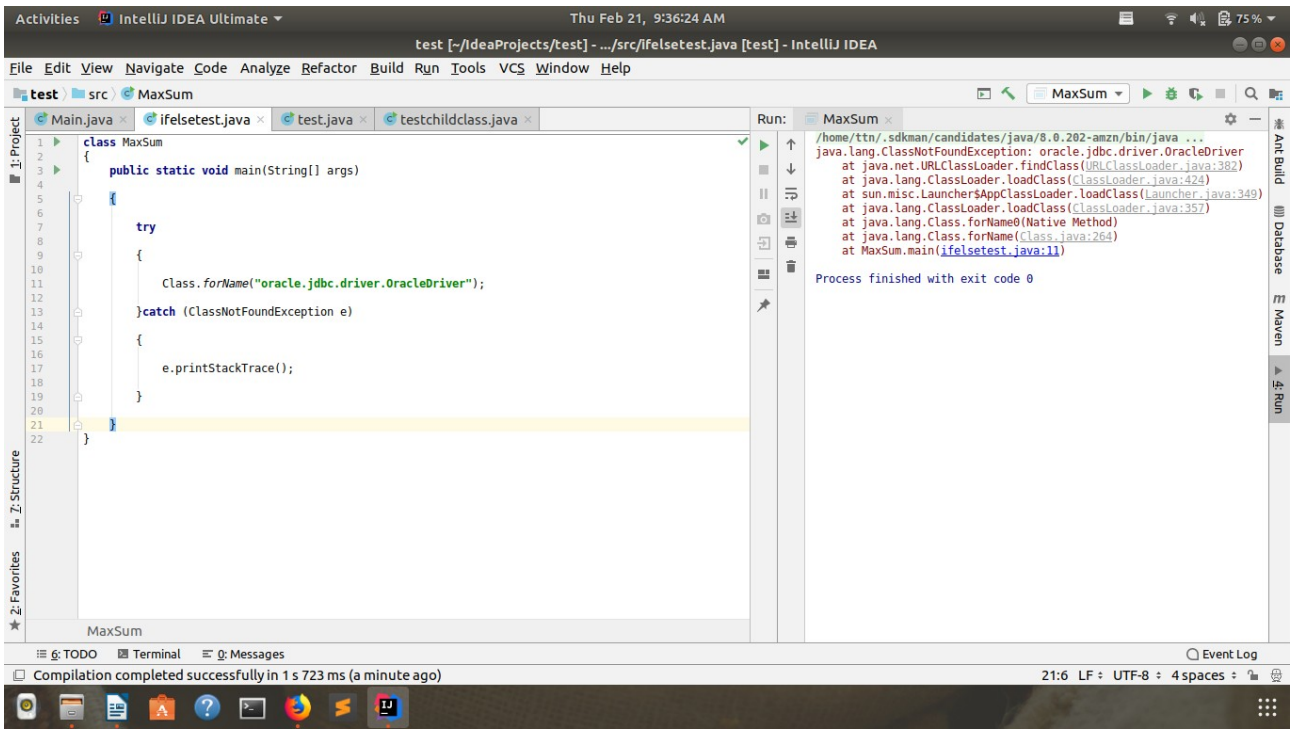
```
        name = sc.next();
        System.out.println("enter the book id : ");
        id = sc.nextInt();
        bookscount++;
        System.out.println("books taken "+bookscount);

    }

    void returnbooks()
    {

        System.out.println("books returned successfully");
        bookscount--;
        System.out.println("number of books taken "+bookscount);



    }


    void showbooks()
    {
        System.out.println("the book name issued is :"+name);
        System.out.println("the book id is :"+id);
    }

}
class lib
{
    public static void main(String args[])
    {
        library lib=new librarian();
        do {
            System.out.println("MENU");
            System.out.println("1.issue books");
            System.out.println("2.return books");
            System.out.println("3.show book details");
            System.out.println("4.exit");
            System.out.println("enter choice :");
            Scanner sc = new Scanner(System.in);
            int choice = sc.nextInt();
            if (choice == 1)
                lib.issuebooks();
            else if (choice == 2)
                lib.returnbooks();
            else if(choice == 3)
                lib.showbooks();
            else
                System.exit(0);
        }while(true);
    }

}
```

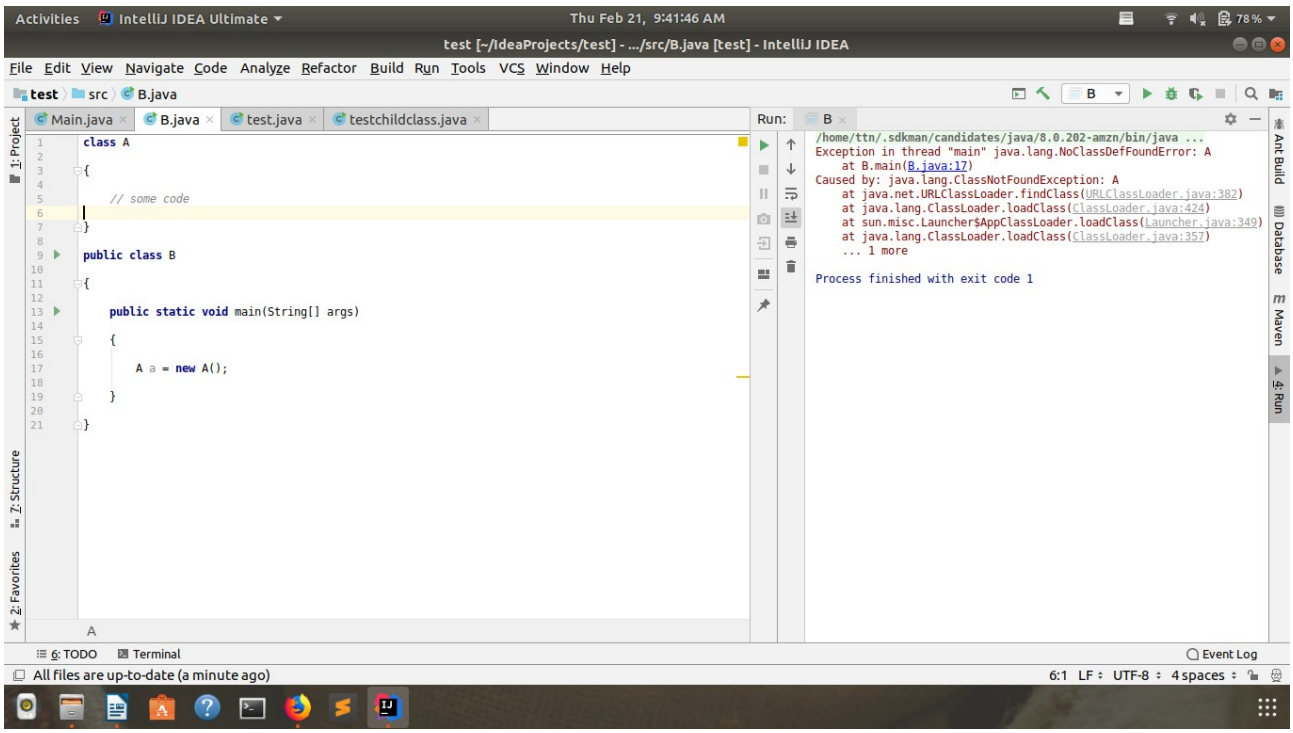
Q2. WAP to sorting string without using string Methods?.
ANS.



Q3.WAP to produce NoClassDefFoundError and ClassNotFoundException exception.

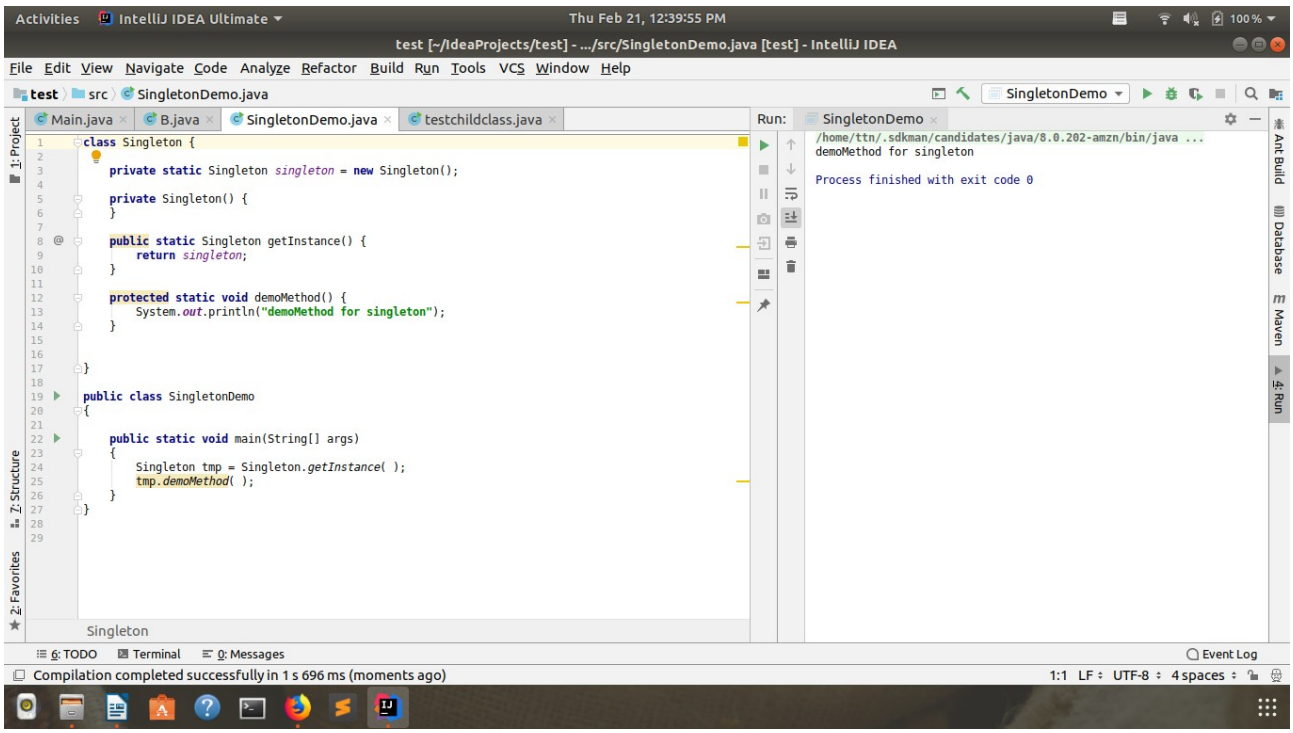
ANS.





Q4. WAP to create singleton class.

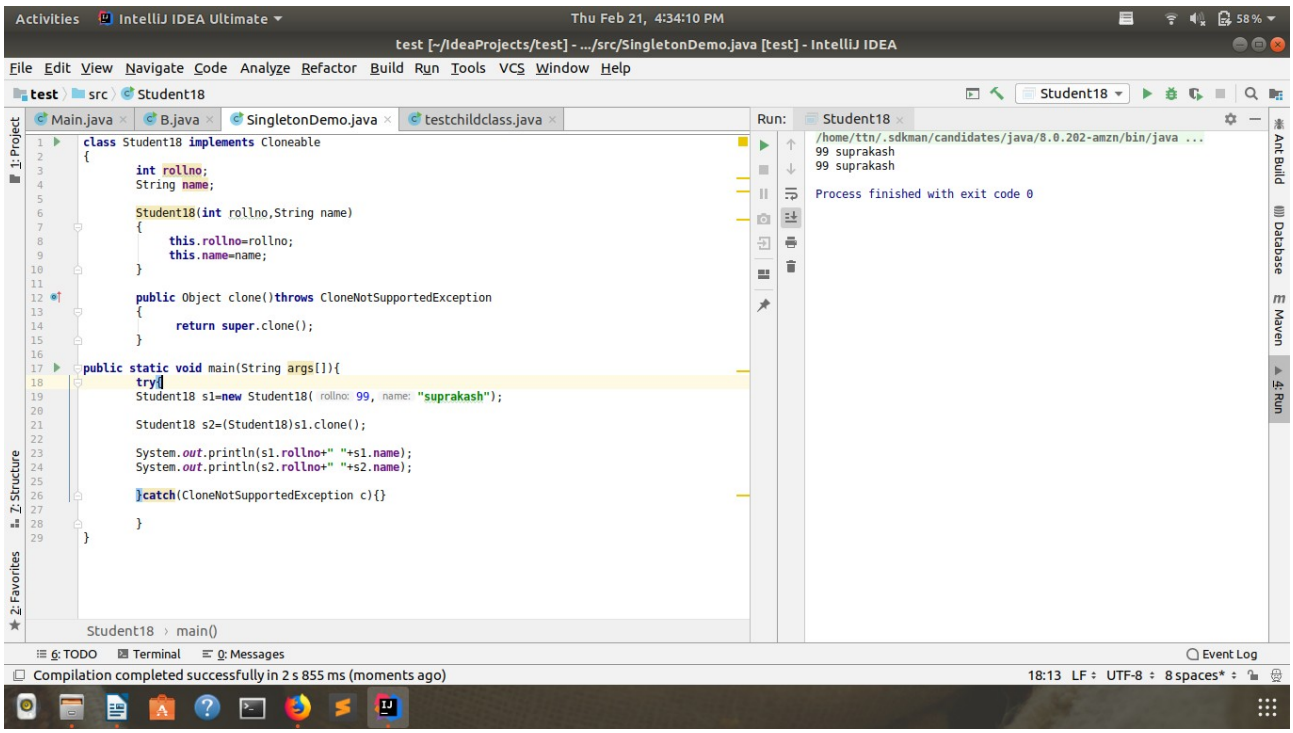
ANS.



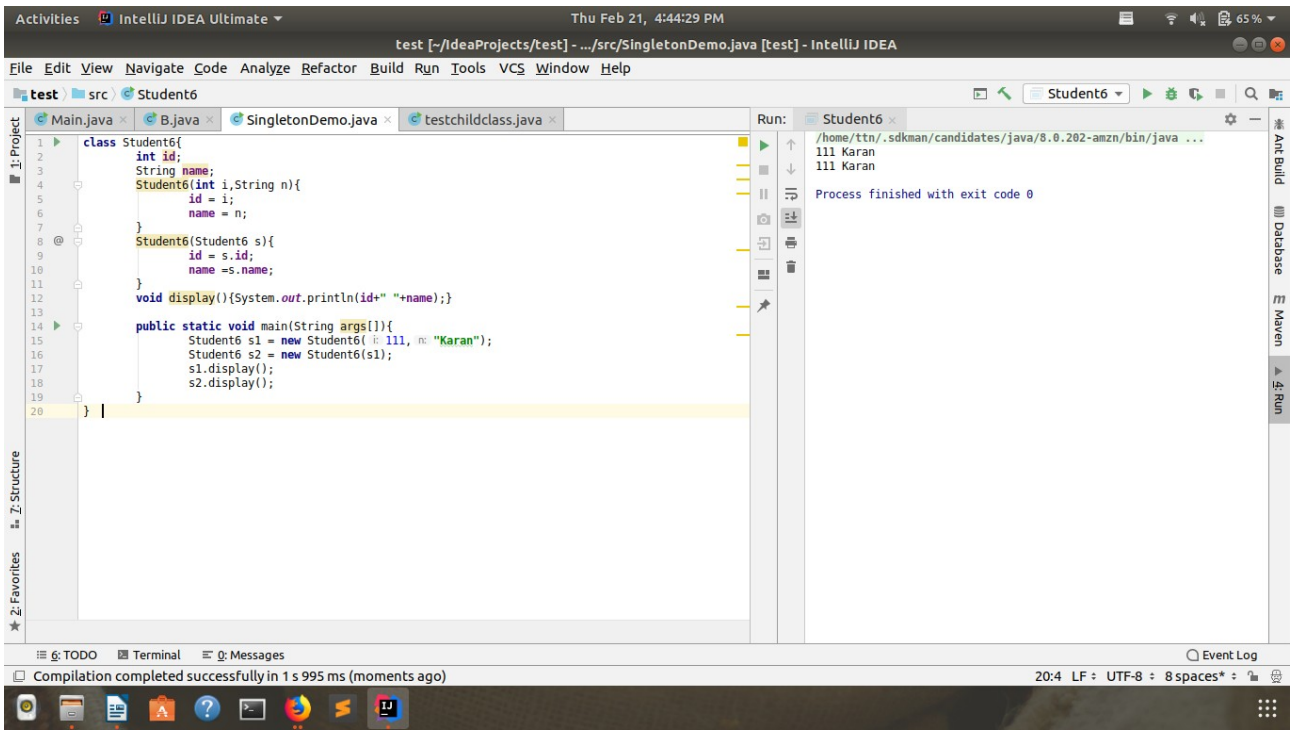
Q5.WAP to show object cloning in java using cloneable and copy constructor both.

ANS.

Using cloneable interface

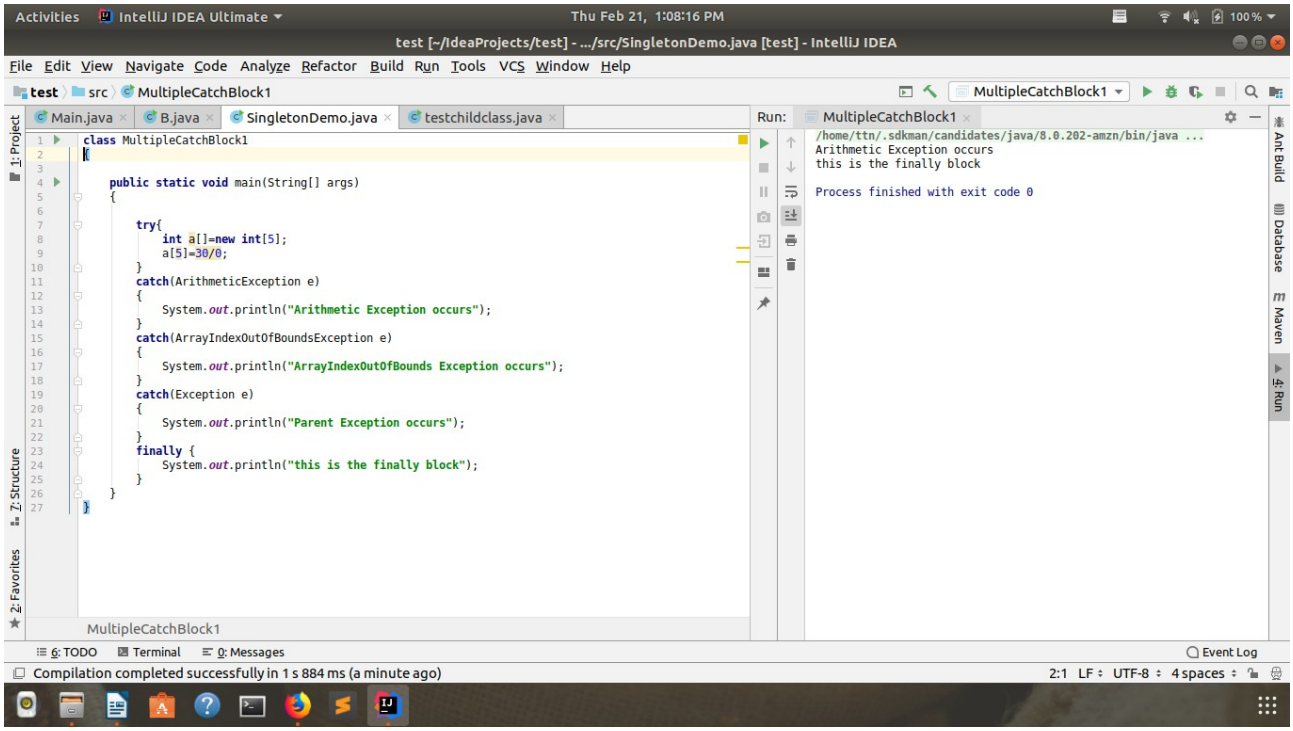


Copy constructor



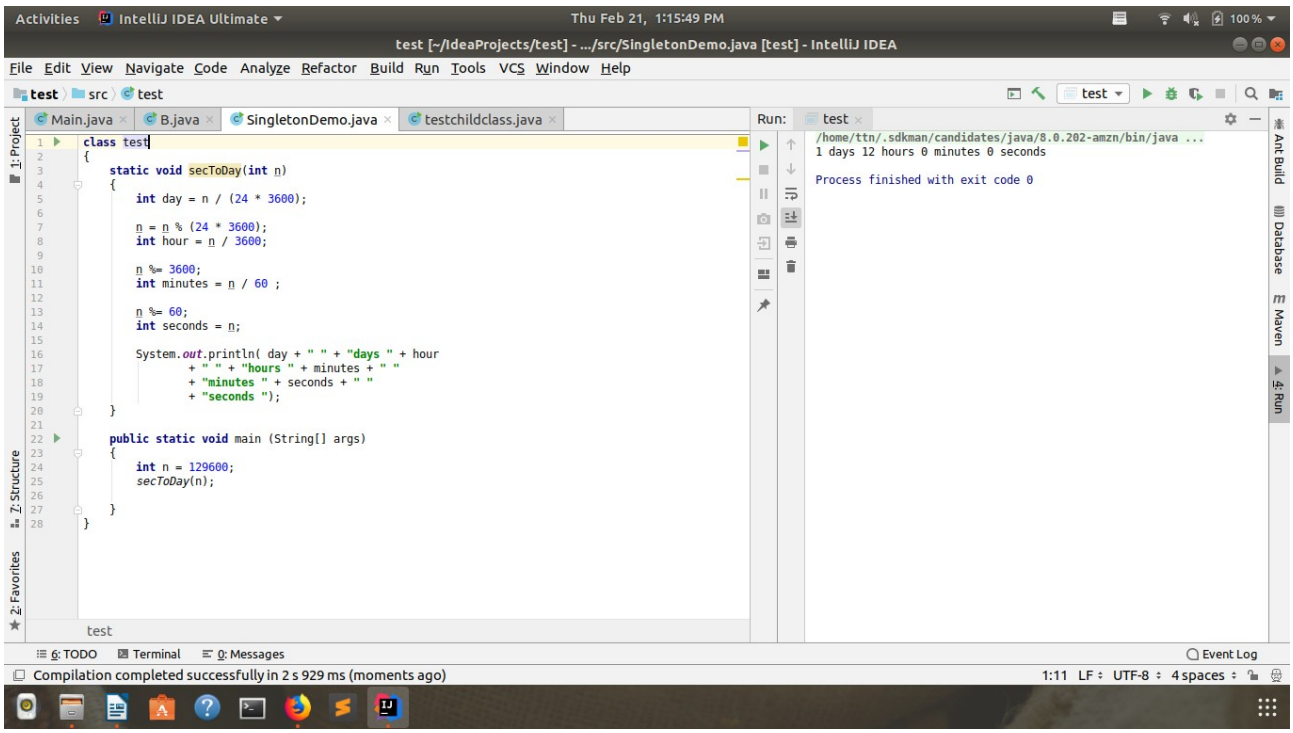
Q6. WAP showing try, multi-catch and finally blocks.

ANS



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

ANS.

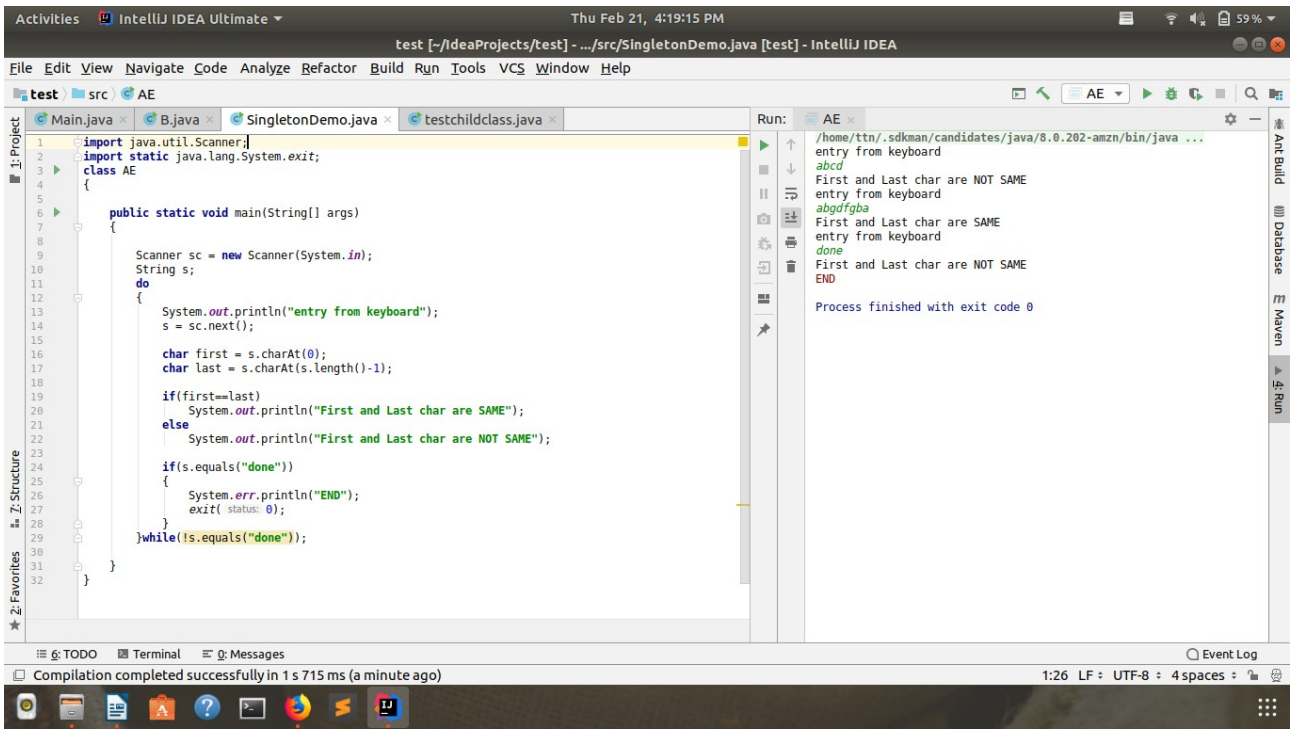


Q8.WAP to read words from the keyboard until the word done is entered. For each word except done, report whether its first character is equal to its last character. For the required loop, use a

a)while statement

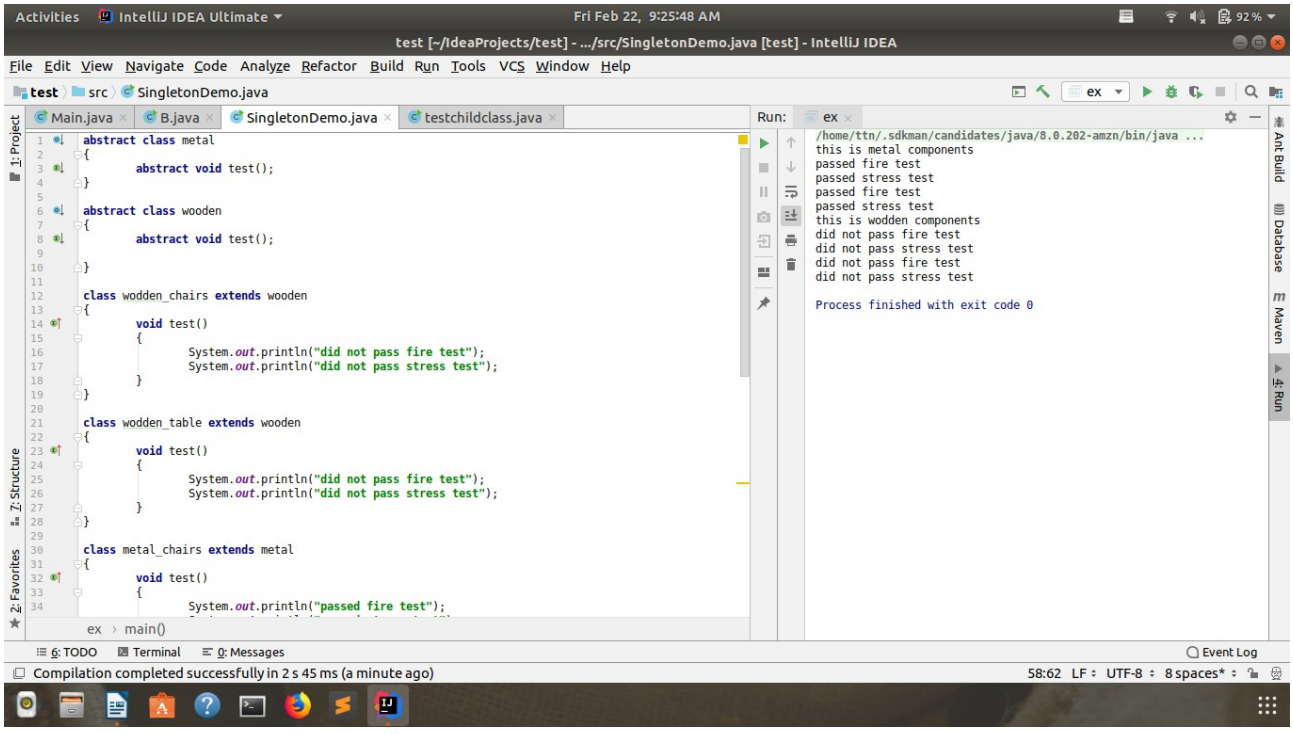
b)do-while statement

ANS.



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.




```
abstract class metal
{
        abstract void test();
}
abstract class wooden
{
        abstract void test();
```

```
}
class wodden_chairs extends wooden
{
        void test()
        {
                System.out.println("did not pass fire test");
                System.out.println("did not pass stress test");
        }
}
class wodden_table extends wooden
{
        void test()
        {
                System.out.println("did not pass fire test");
                System.out.println("did not pass stress test");
        }
}
class metal_chairs extends metal
{
        void test()
        {
                System.out.println("passed fire test");
                System.out.println("passed stress test");
        }
}
class metal_table extends metal
{
        void test()
        {
                System.out.println("passed fire test");
                System.out.println("passed stress test");
        }

}
class ex
{
        public static void main(String args[])
        {
                metal c = new metal_chairs();
                metal t = new metal_table();
                System.out.println("this is metal components");
                c.test();
                t.test();
                System.out.println("this is wodden components");
                wooden cc =new wodden_chairs();
                wooden tt =new wodden_table();
                cc.test();
                tt.test();
        }

}
```

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

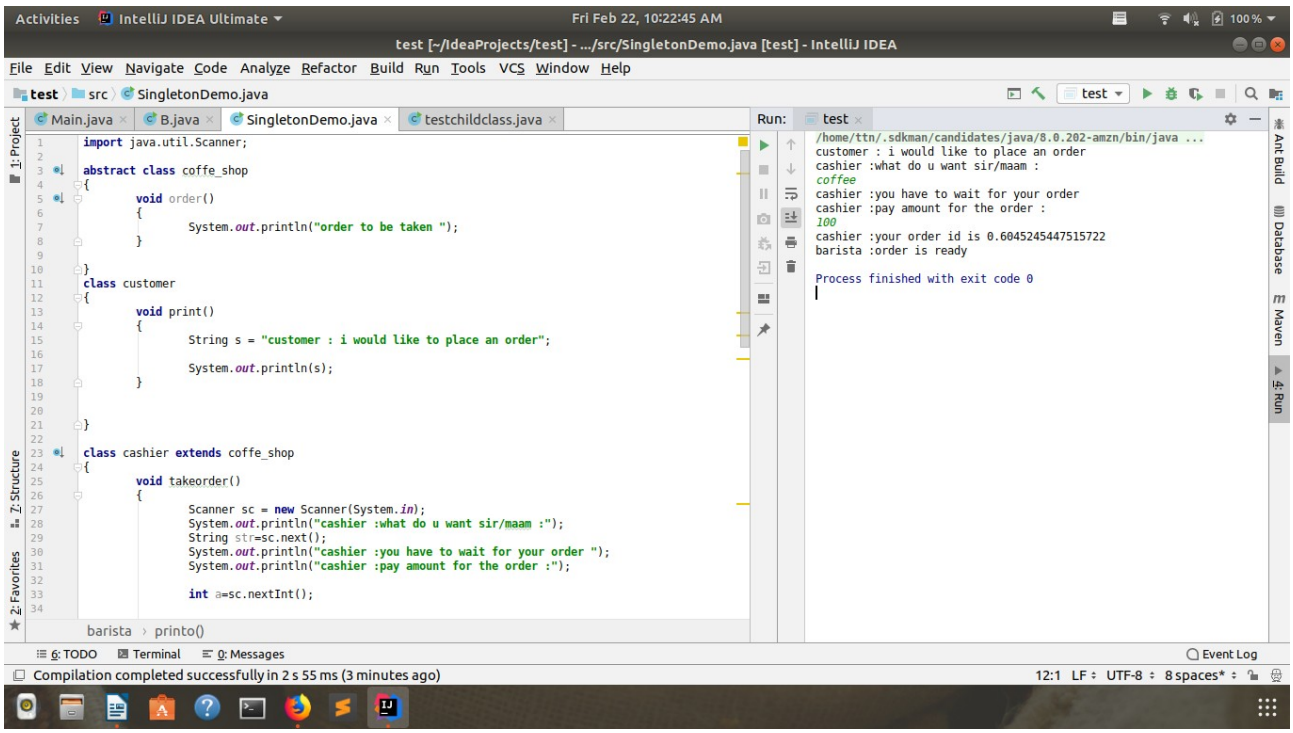
* Customer
  - Pays the cash to the cashier and places his order, get a token number back
  - Waits for the intimation that order for his token is ready
  - Upon intimation/notification he collects the coffee and enjoys his drink
  (Assumption: Customer waits till the coffee is done, he wont timeout 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.



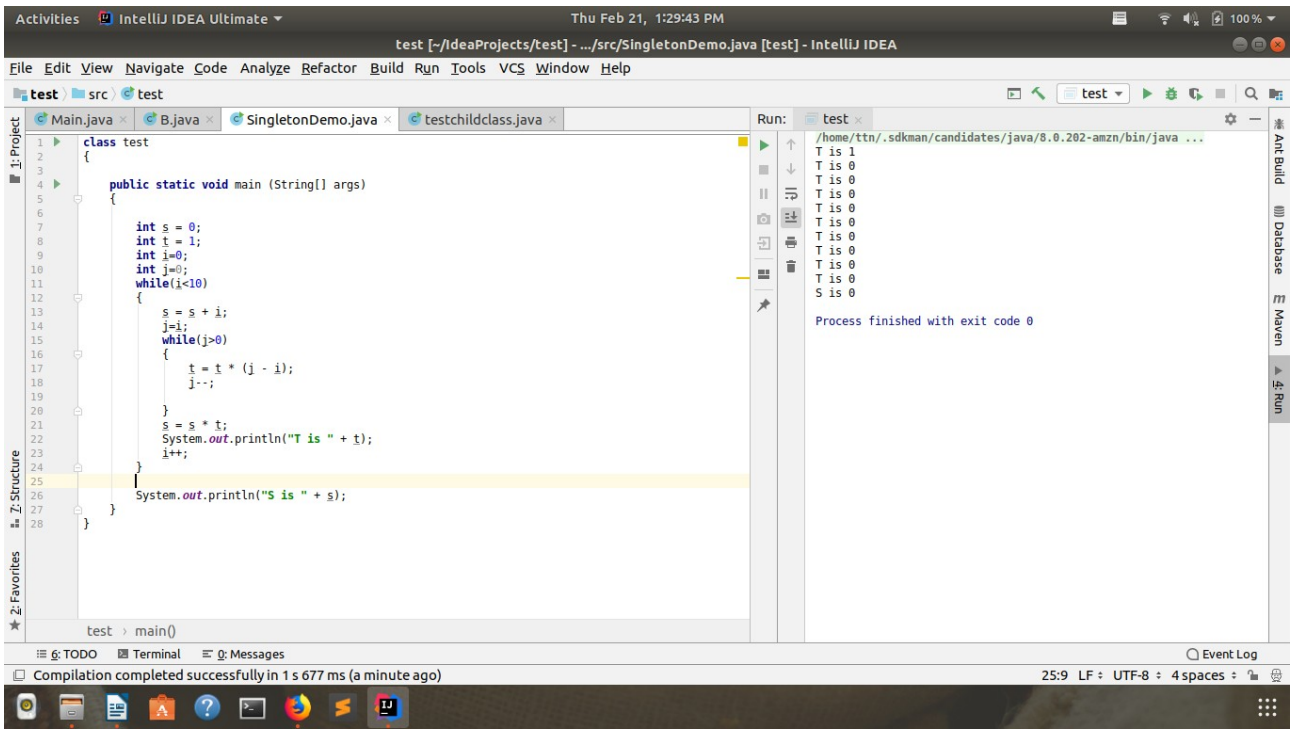

```java
import java.util.Scanner;
abstract class coffe_shop
{
        void order()
        {
                System.out.println("order to be taken ");
        }
}
class customer
{
        void print()
        {
                String s = "customer : i would like to place an order";
                System.out.println(s);
        }


}
class cashier extends coffe_shop
{
        void takeorder()
        {
                Scanner sc = new Scanner(System.in);
                System.out.println("cashier :what do u want sir/maam :");
                String str=sc.next();
                System.out.println("cashier :you have to wait for your order ");
                System.out.println("cashier :pay amount for the order :");

                int a=sc.nextInt();


        }
        double a = Math.random();
        void order()
        {
                System.out.println("cashier :your order id is " +a);
                //sc = new Scanner(System.in);
```

```
        }

}
class barista extends cashier
{
        void printo()
        {
                System.out.println("barista :order is ready  ");

        }


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

                customer c=new customer();
                cashier cash=new cashier();
                barista bb=new barista();

                c.print();
                cash.takeorder();
                cash.order();

                bb.printo();

        }
}
```

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.

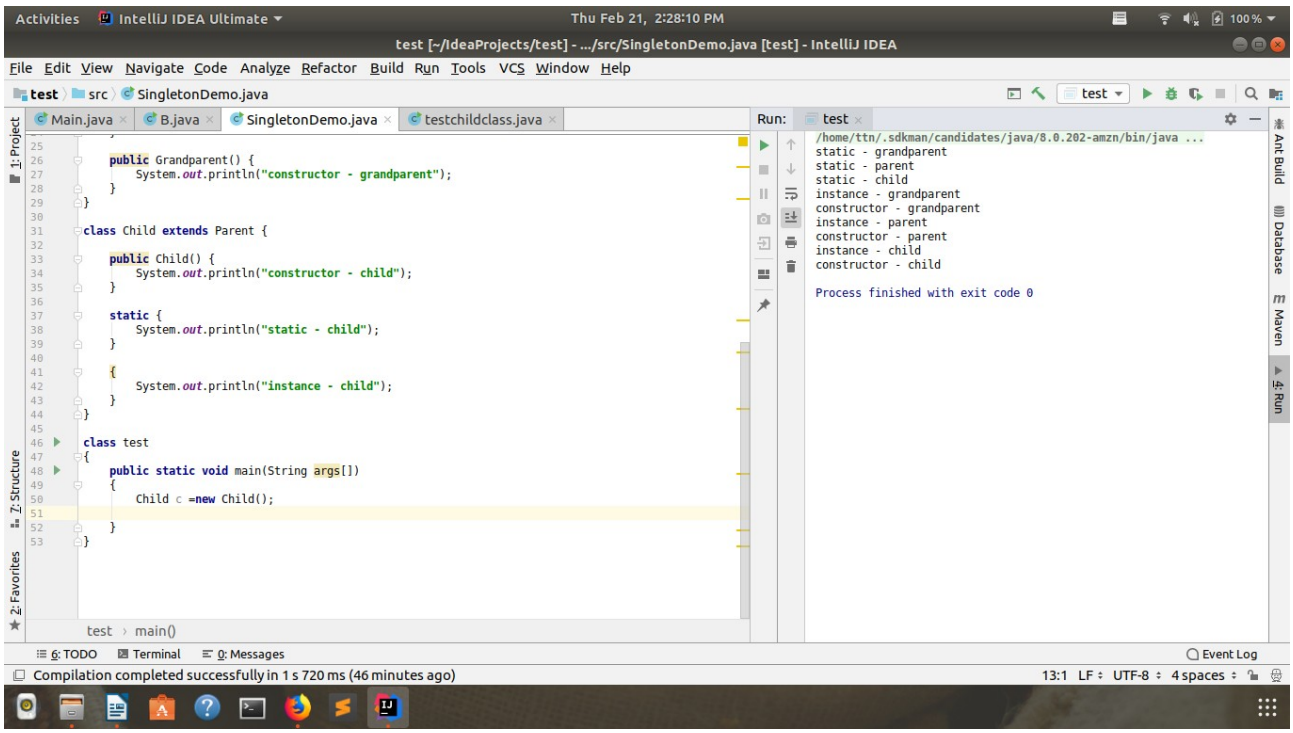


12.What will be the output on new Child(); ?

```
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");
    }
}
```

ANS.



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

ANS.