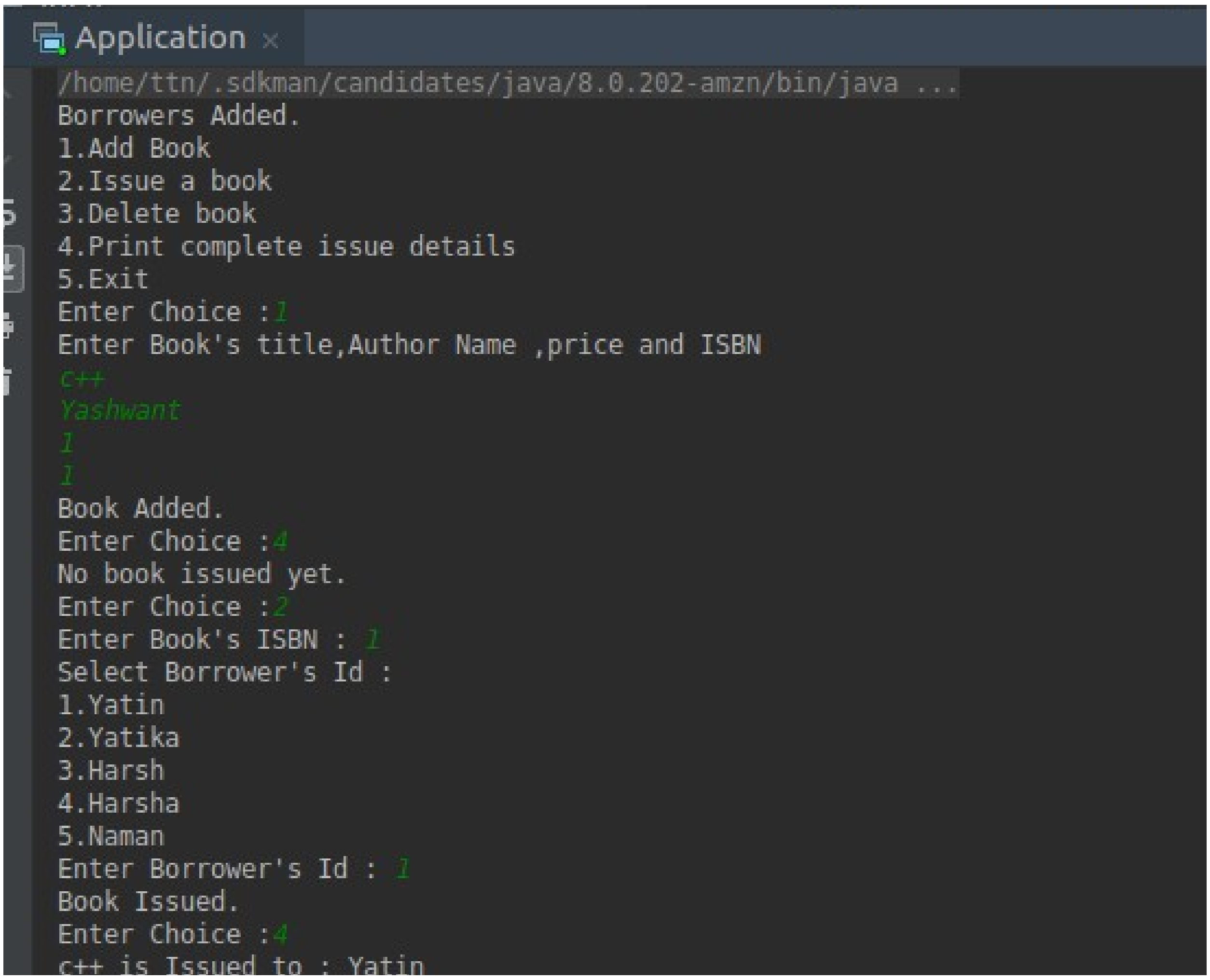
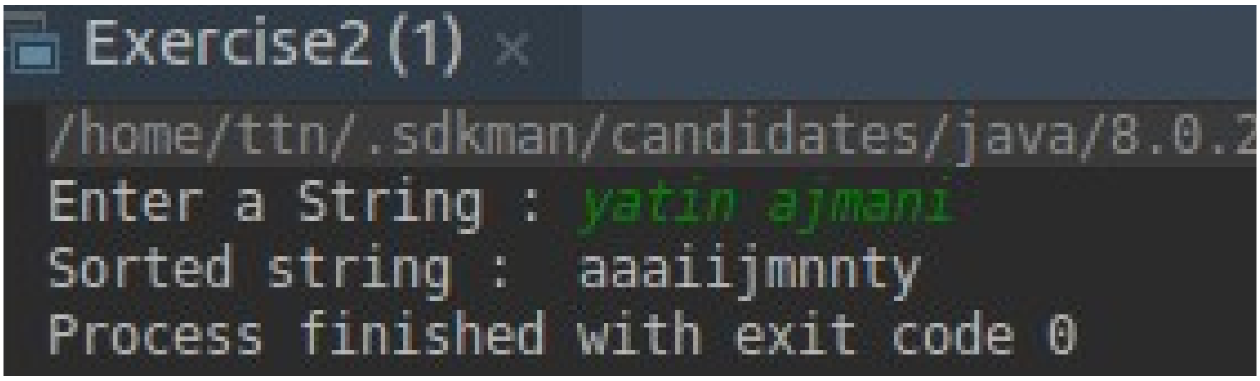
Exercise

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.



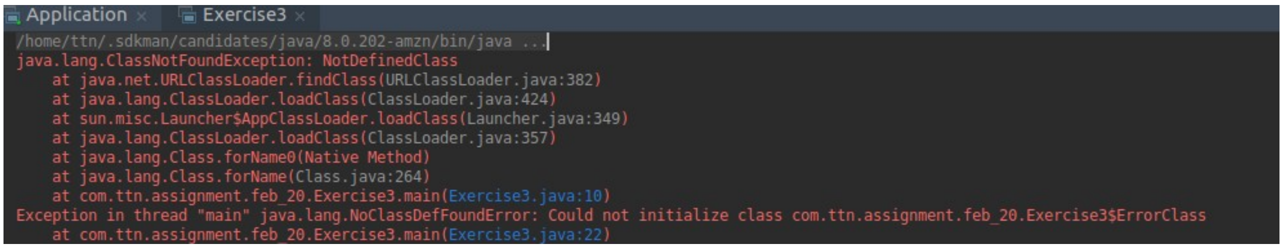
```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Borrowers Added.
1.Add Book
2.Issue a book
3.Delete book
4.Print complete issue details
5.Exit
Enter Choice :1
Enter Book's title,Author Name ,price and ISBN
c++
Yashwant
1
1
Book Added.
Enter Choice :4
No book issued yet.
Enter Choice :2
Enter Book's ISBN : 1
Select Borrower's Id :
1.Yatin
2.Yatika
3.Harsh
4.Harsha
5.Naman
Enter Borrower's Id : 1
Book Issued.
Enter Choice :4
c++ is Issued to : Yatin
```

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



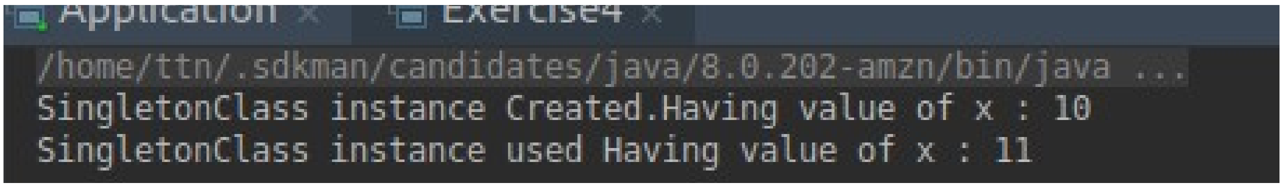
```
/home/ttn/.sdkman/candidates/java/8.0.2
Enter a String : yatin ajmani
Sorted string :  aaaiijmnnty
Process finished with exit code 0
```

3. WAP to produce NoClassDefFoundError and ClassNotFoundException exception.



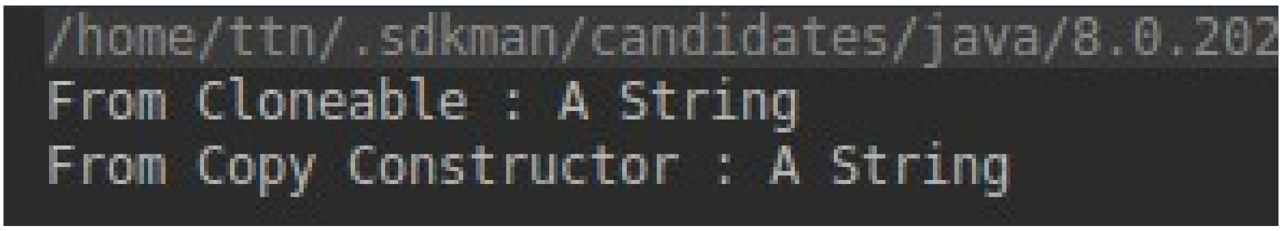
```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
java.lang.ClassNotFoundException: NotDefinedClass
    at java.net.URLClassLoader.findClass(URLClassLoader.java:382)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:349)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Class.java:264)
    at com.ttn.assignment.feb_20.Exercise3.main(Exercise3.java:10)
Exception in thread "main" java.lang.NoClassDefFoundError: Could not initialize class com.ttn.assignment.feb_20.Exercise3$ErrorClass
    at com.ttn.assignment.feb_20.Exercise3.main(Exercise3.java:22)
```

4. WAP to create singleton class.



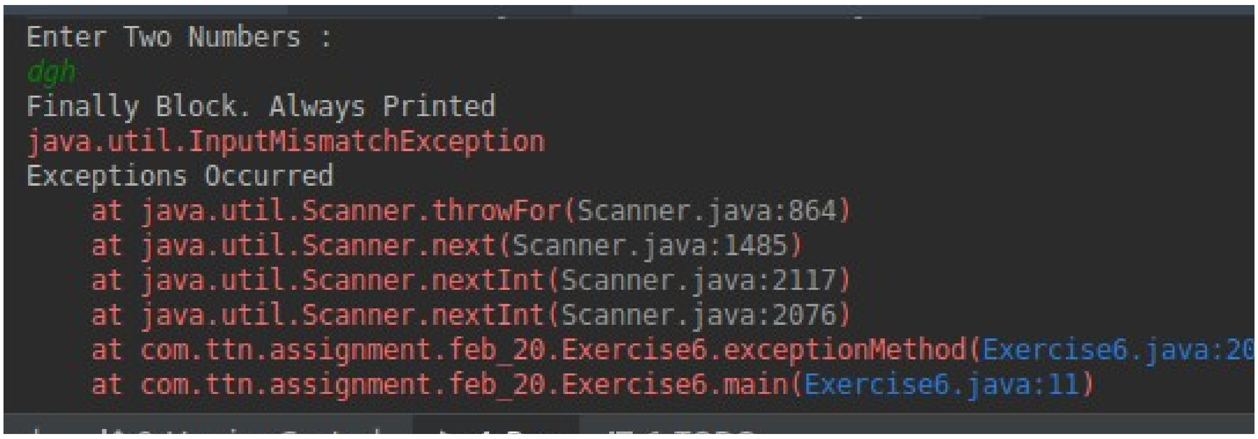
```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
SingletonClass instance Created.Having value of x : 10
SingletonClass instance used Having value of x : 11
```

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



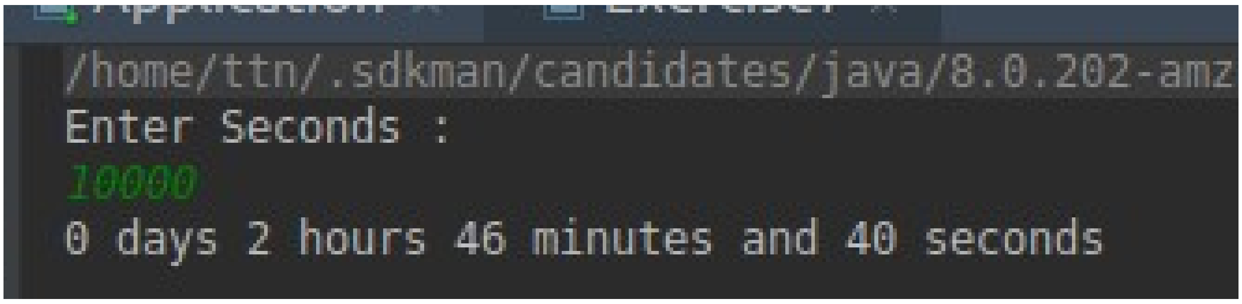
```
/home/ttn/.sdkman/candidates/java/8.0.202
From Cloneable : A String
From Copy Constructor : A String
```

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



```
Enter Two Numbers :
dgh
Finally Block. Always Printed
java.util.InputMismatchException
Exceptions Occurred
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at com.ttn.assignment.feb_20.Exercise6.exceptionMethod(Exercise6.java:20
    at com.ttn.assignment.feb_20.Exercise6.main(Exercise6.java:11)
```

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

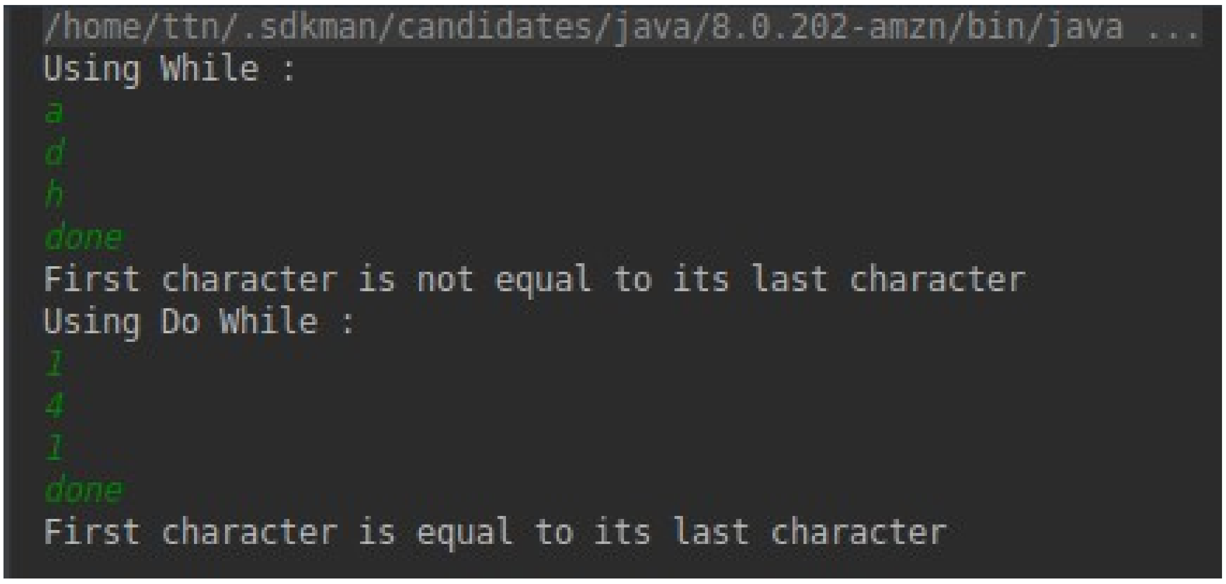


```
/home/ttn/.sdkman/candidates/java/8.0.202-amz
Enter Seconds :
10000
0 days 2 hours 46 minutes and 40 seconds
```

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

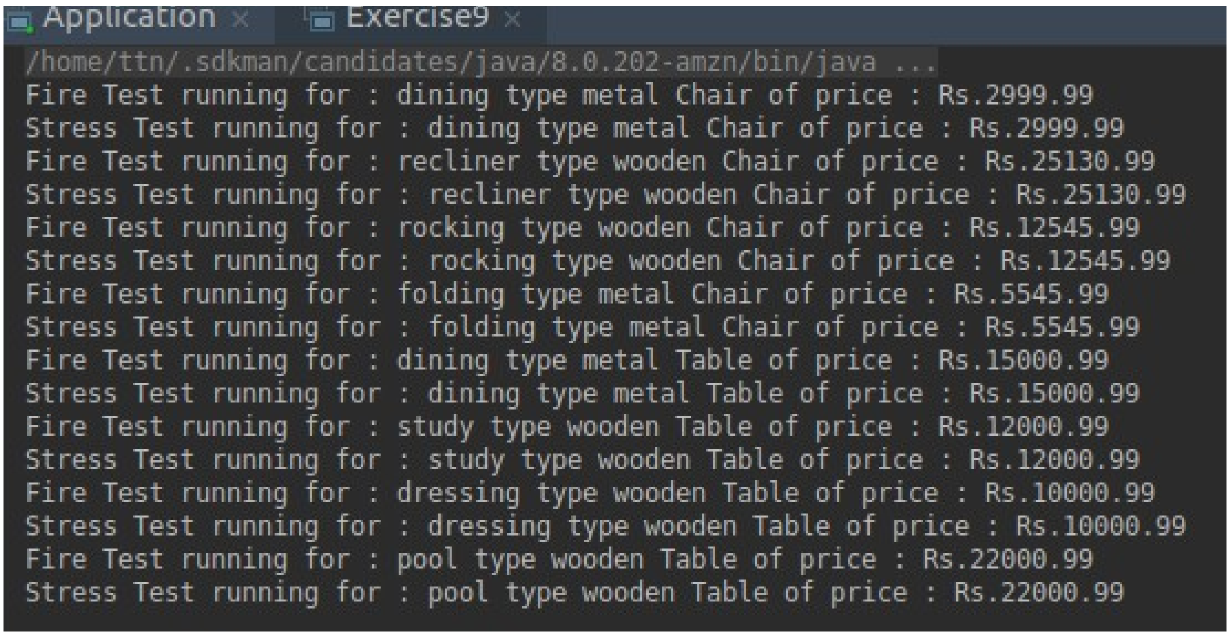
a)while statement

b)do-while statement



```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Using While :
a
d
h
done
First character is not equal to its last character
Using Do While :
1
4
1
done
First character is equal to its last character
```

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.



```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Fire Test running for : dining type metal Chair of price : Rs.2999.99
Stress Test running for : dining type metal Chair of price : Rs.2999.99
Fire Test running for : recliner type wooden Chair of price : Rs.25130.99
Stress Test running for : recliner type wooden Chair of price : Rs.25130.99
Fire Test running for : rocking type wooden Chair of price : Rs.12545.99
Stress Test running for : rocking type wooden Chair of price : Rs.12545.99
Fire Test running for : folding type metal Chair of price : Rs.5545.99
Stress Test running for : folding type metal Chair of price : Rs.5545.99
Fire Test running for : dining type metal Table of price : Rs.15000.99
Stress Test running for : dining type metal Table of price : Rs.15000.99
Fire Test running for : study type wooden Table of price : Rs.12000.99
Stress Test running for : study type wooden Table of price : Rs.12000.99
Fire Test running for : dressing type wooden Table of price : Rs.10000.99
Stress Test running for : dressing type wooden Table of price : Rs.10000.99
Fire Test running for : pool type wooden Table of price : Rs.22000.99
Stress Test running for : pool type wooden Table of price : Rs.22000.99
```

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)

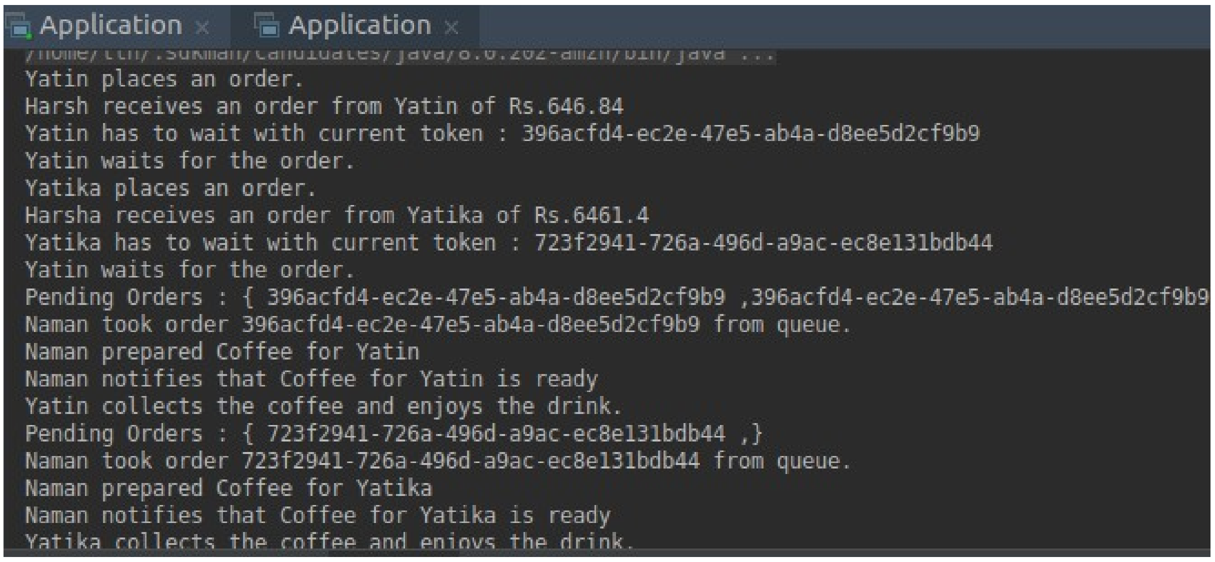
* Barista

  - 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



```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Yatin places an order.
Harsh receives an order from Yatin of Rs.646.84
Yatin has to wait with current token : 396acfd4-ec2e-47e5-ab4a-d8ee5d2cf9b9
Yatin waits for the order.
Yatika places an order.
Harsha receives an order from Yatika of Rs.6461.4
Yatika has to wait with current token : 723f2941-726a-496d-a9ac-ec8e131bdb44
Yatin waits for the order.
Pending Orders : { 396acfd4-ec2e-47e5-ab4a-d8ee5d2cf9b9 ,396acfd4-ec2e-47e5-ab4a-d8ee5d2cf9b9
Naman took order 396acfd4-ec2e-47e5-ab4a-d8ee5d2cf9b9 from queue.
Naman prepared Coffee for Yatin
Naman notifies that Coffee for Yatin is ready
Yatin collects the coffee and enjoys the drink.
Pending Orders : { 723f2941-726a-496d-a9ac-ec8e131bdb44 ,}
Naman took order 723f2941-726a-496d-a9ac-ec8e131bdb44 from queue.
Naman prepared Coffee for Yatika
Naman notifies that Coffee for Yatika is ready
Yatika collects the coffee and enjoys the drink.
```

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);
```

Ans.

```
int s = 0;

int t = 1;
int i = 0;
while (i < 10) {
    s = s + i;
    int 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);
```

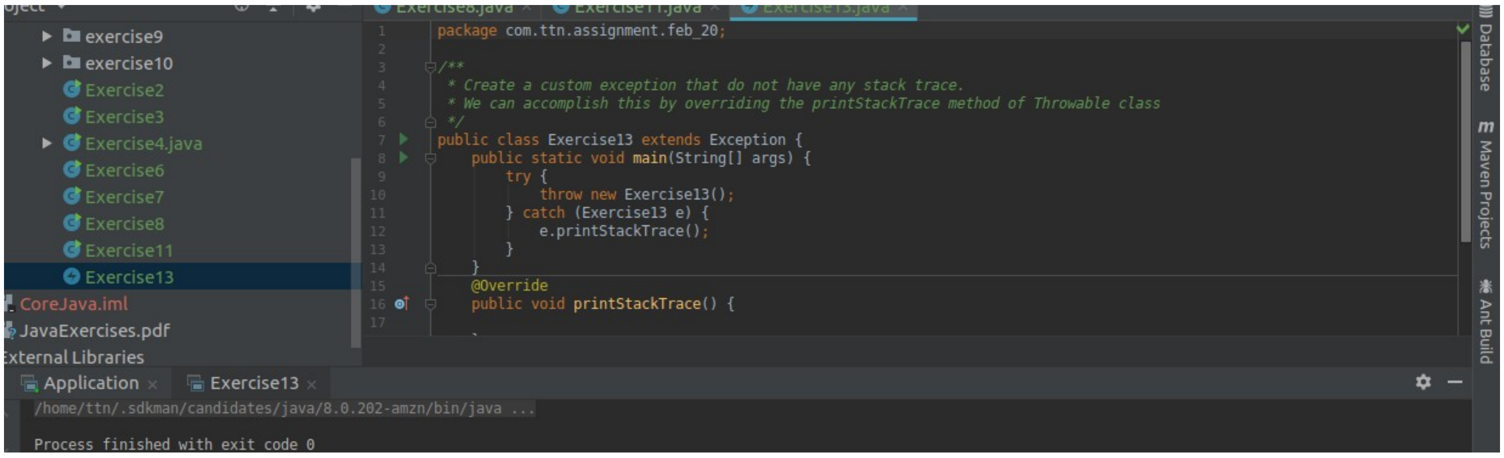
12.What will be the output on new Child(); ?

```
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. Firstly the static blocks will be executed from the grandparent following hierarchy,then the child class construct will implicitly call parent’s constructor and parent’s constructor will implicitly call grandparent’s constructor hence the instance initialization block of the grandparent will be executed as it is always executed before the constructor then the constructor of grandparent and then following same hierarchy for children classes.

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



```
package com.ttn.assignment.feb_20;

/**
 * Create a custom exception that do not have any stack trace.
 * We can accomplish this by overriding the printStackTrace method of Throwable class
 */
public class Exercise13 extends Exception {
    public static void main(String[] args) {
        try {
            throw new Exercise13();
        } catch (Exercise13 e) {
            e.printStackTrace();
        }
    }
    @Override
    public void printStackTrace() {

```

```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...

Process finished with exit code 0
```