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.

```
import java.util.ArrayList;




interface  Person {
   void getPerson();
}










class Book {
   String bookName;
   String bookAuthor;
```

```
    float bookPrice;

    Book(String name, String author, float price) {
        bookName = name;
        bookAuthor = author;
        bookPrice = price;
    }

    void getBookDetails() {
        System.out.println("Name: " + bookName);
        System.out.println("Author: " + bookAuthor);
        System.out.println("Price: " + bookPrice);
    }
}
```

```
class Librarian implements Person {
    String librarianName;
    int librarianID;

    Librarian(String name, int id) {
        this.librarianName = name;
        this.librarianID = id;
```

```java
    }

    // Interface method definition
    public void getPerson() {
        System.out.println("Name: " + librarianName);
        System.out.println("ID: " + librarianID);
    }
}



class Member implements Person {
    String memberName;
    int memberID;

    Member(String name, int id) {
        memberName = name;
        memberID = id;
    }

    // Interface method definition
    public void getPerson() {
        System.out.println("Name: " + memberName);
        System.out.println("ID: " + memberID);
    }
```

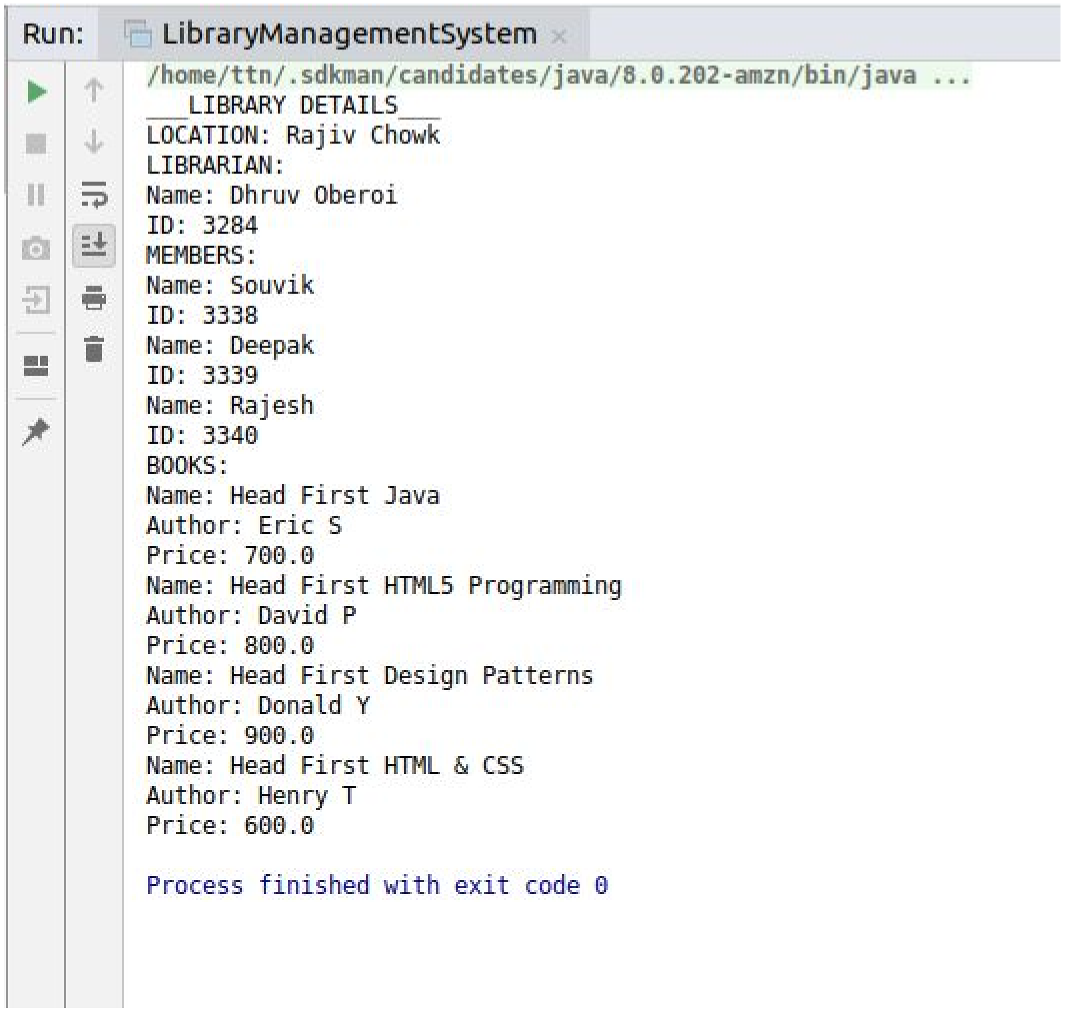
```
}
```

```
class Library {
   private String libraryLocation;
   private ArrayList<Book> booksInLibrary;
   private ArrayList<Member> membersOfLibrary;
   private Librarian librarian;

   public String getLibraryLocation() {
      return libraryLocation;
   }

   public void setLibraryLocation(String libraryLocation) {
      this.libraryLocation = libraryLocation;
   }

   public ArrayList<Book> getBooksInLibrary() {
      return booksInLibrary;
   }
   public void setBooksInLibrary(ArrayList<Book>
booksInLibrary) {
      this.booksInLibrary = booksInLibrary;
   }

   public ArrayList<Member> getMembersOfLibrary() {
      return membersOfLibrary;
   }

   public void setMembersOfLibrary(ArrayList<Member>
membersOfLibrary) {
      this.membersOfLibrary = membersOfLibrary;
   }

   public Librarian getLibrarian() {
      return librarian;
   }

   public void setLibrarian(Librarian librarian) {
      this.librarian = librarian;
   }

   void getLibraryDetails() {
      System.out.println("___LIBRARY DETAILS___");
      System.out.println("LOCATION: " +
libraryLocation);
```

```
        System.out.println("LIBRARIAN:");
        librarian.getPerson();
        System.out.println("MEMBERS:");
        for (Member m : membersOfLibrary)
            m.getPerson();
        System.out.println("BOOKS:");
        for (Book b : booksInLibrary)
            b.getBookDetails();
    }
}
```

```
public class LibraryManagementSystem {
    public static void main(String[] args) {
        Library myLibrary = new Library();
        myLibrary.setLibraryLocation("Rajiv Chowk");
        Book b1 = new Book("Head First Java", "Eric S",
700.0f);
        Book b3 = new Book("Head First Design Patterns",
"Donald Y", 900.0f);
        Book b2 = new Book("Head First HTML5
Programming", "David P", 800.0f);
        Book b4 = new Book("Head First HTML & CSS",
"Henry T", 600.0f);
```

```
        ArrayList<Book> bookList = new ArrayList<>();
        bookList.add(b1);
        bookList.add(b2);
        bookList.add(b3);
        bookList.add(b4);
        myLibrary.setBooksInLibrary(bookList);
        Librarian librarian = new Librarian("Dhruv Oberoi",
3284);
        myLibrary.setLibrarian(librarian);
        Member m1 = new Member("Souvik", 3338);
        Member m2 = new Member("Deepak", 3339);
        Member m3 = new Member("Rajesh", 3340);
        ArrayList<Member> memberList = new ArrayList<>();
        memberList.add(m1);
        memberList.add(m2);
        memberList.add(m3);
        myLibrary.setMembersOfLibrary(memberList);
        myLibrary.getLibraryDetails();
    }
}
```

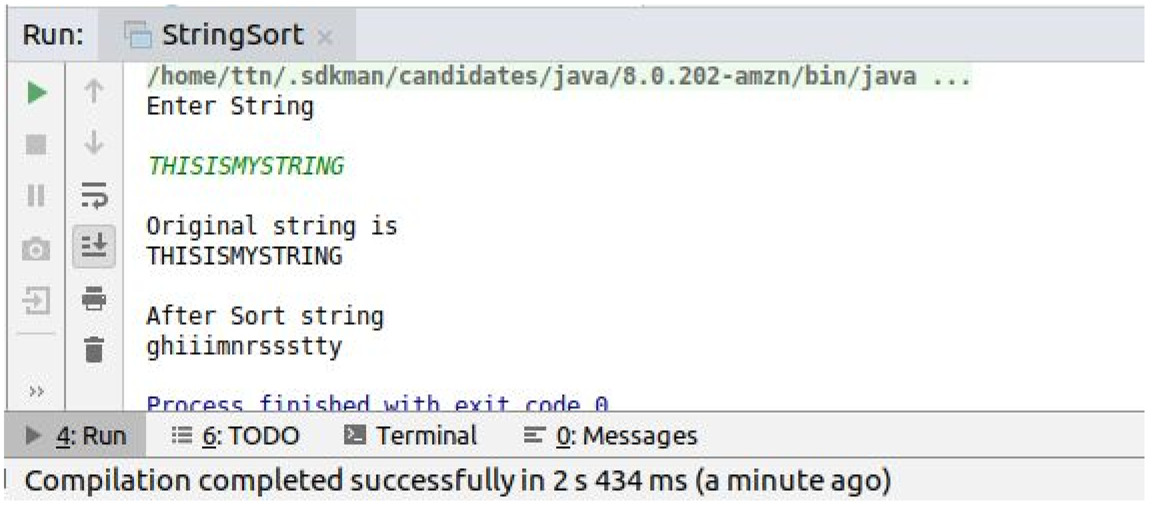


```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
___LIBRARY DETAILS___
LOCATION: Rajiv Chowk
LIBRARIAN:
Name: Dhruv Oberoi
ID: 3284
MEMBERS:
Name: Souvik
ID: 3338
Name: Deepak
ID: 3339
Name: Rajesh
ID: 3340
BOOKS:
Name: Head First Java
Author: Eric S
Price: 700.0
Name: Head First HTML5 Programming
Author: David P
Price: 800.0
Name: Head First Design Patterns
Author: Donald Y
Price: 900.0
Name: Head First HTML & CSS
Author: Henry T
Price: 600.0

Process finished with exit code 0
```

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

```
import java.util.Scanner;
public class StringSort{
  public static void main(String[] args){
      Scanner sc= new Scanner(System.in);
```

```
        System.out.println("Enter String\n");
        String input=sc.next();
        System.out.println("\nOriginal string is\n"+input);
        String output=sortString(input);
        System.out.println("\nAfter Sort string\n"+output);
    }
    public static String sortString(String input){
        char[] charArray=new char[input.length()];
        input=input.toLowerCase();
        int index=0;
        for(int i='a';i<='z';i++){
            for(int j=0;j<input.length();j++){
                if(input.charAt(j)==i)
                    charArray[index++]=(char)i;
            }
        }
        return new String(charArray);
    }
}
```

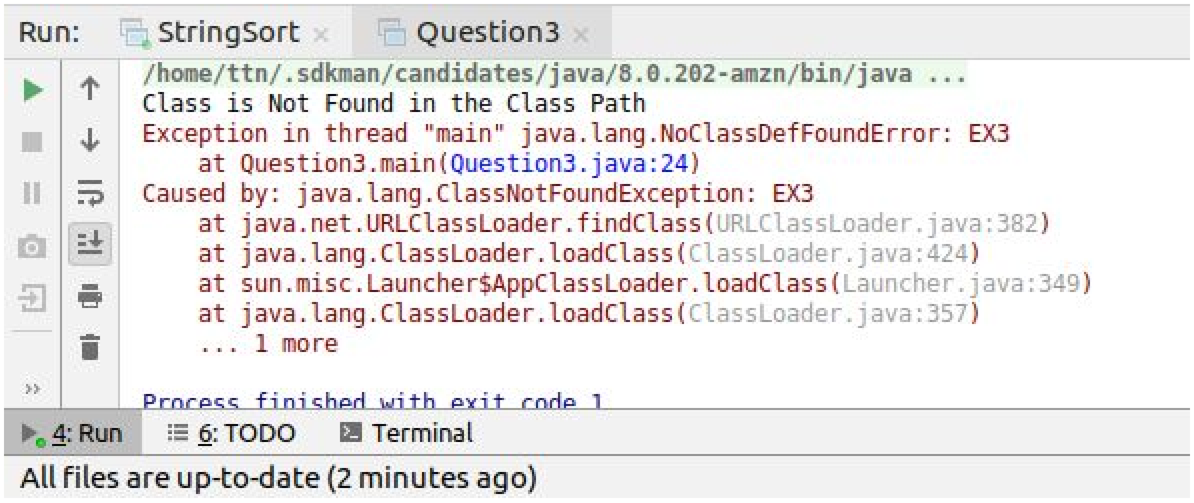


```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Enter String

THISISMYSTRING

Original string is
THISISMYSTRING

After Sort string
ghiiimnrssstty

Process finished with exit code 0
```

```
Compilation completed successfully in 2 s 434 ms (a minute ago)
```

Q3.WAP to produce NoClassDefFoundError and ClassNotFoundException exception.

```
class EX3
{
  void test1()
  {
     System.out.println("this is test1 method");
  }
}



public class Question3
{

  public static void main(String args[]) {
        try {
            Class.forName("Dhruv");


        } catch (ClassNotFoundException ex) {
            System.out.println("Class is Not Found in the
Class Path");
            //ex.printStackTrace();
        }

        EX3 ob = new EX3();

    }
}
```

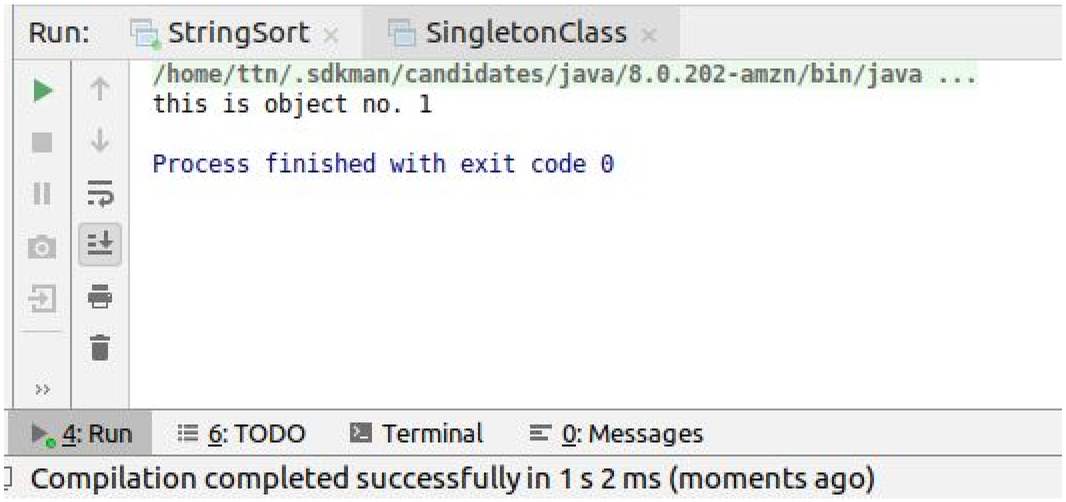


```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Class is Not Found in the Class Path
Exception in thread "main" java.lang.NoClassDefFoundError: EX3
	at Question3.main(Question3.java:24)
Caused by: java.lang.ClassNotFoundException: EX3
	at java.net.URLClassLoader.findClass(URLClassLoader.java:382)
	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)
	... 1 more

Process finished with exit code 1
```

All files are up-to-date (2 minutes ago)

4. WAP to create singleton class.

```
class SingleTest{
    private static SingleTest t=null;

    private SingleTest(){
    }

    static int i=0;
    public static SingleTest getSingleTest()
    {
        if(t==null){
            t=new SingleTest();
            System.out.println("this is object no. "+ (++i));

        }
        return t;
    }
}



public class SingletonClass {

    public static void main(String[] args) {
        SingleTest t1=SingleTest.getSingleTest();
        SingleTest t2=SingleTest.getSingleTest();
    }
}
```



```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
this is object no. 1

Process finished with exit code 0
```

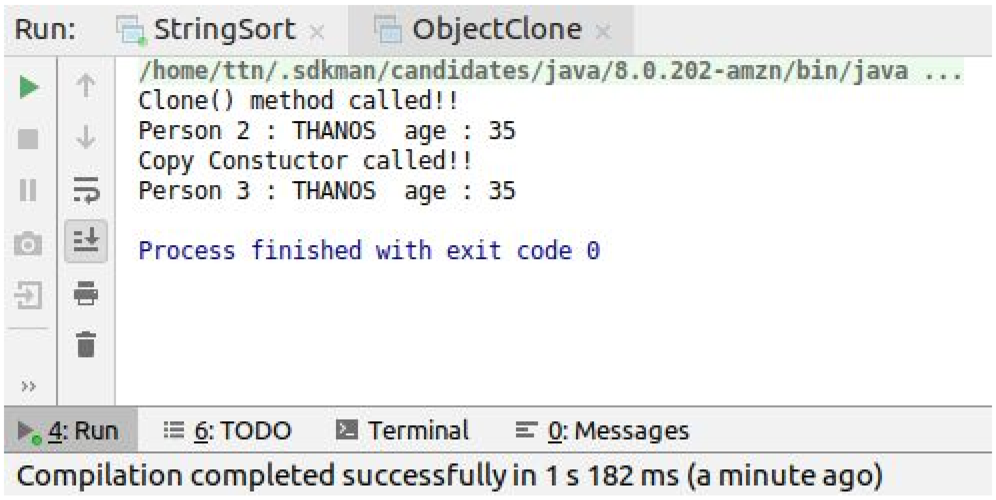
Compilation completed successfully in 1 s 2 ms (moments ago)

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

```
class Person1 implements Cloneable
{
  String Name;
  int Age;

  Person1(String Name, int Age)
  {
     this.Name = Name;
     this.Age = Age;
  }

  Person1(Person1 s)
  {
     System.out.println("Copy Constuctor called!!");
```

```
      Name = s.Name;
      Age = s.Age;
   }

   public Object clone()throws
CloneNotSupportedException
   {
      System.out.println("Clone() method called!!");
      return super.clone();
   }

}

public class ObjectClone
{
   public static void main(String[] args)
   {
      try
      {
         Person1 s1 = new Person1("THANOS", 35);
         Person1 s2 = (Person1) s1.clone();
         System.out.println("Person 2 : "+s2.Name+"
age : "+s2.Age);
         Person1 s3 = new Person1(s1);
         System.out.println("Person 3 : "+s3.Name+"
age : "+s3.Age);
```

```
        }
        catch (CloneNotSupportedException e)
        {
            System.out.println("Exception: "+e);
        }
    }

}
```



```
Run:   StringSort ×   ObjectClone ×
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Clone() method called!!
Person 2 : THANOS  age : 35
Copy Constuctor called!!
Person 3 : THANOS  age : 35

Process finished with exit code 0
```

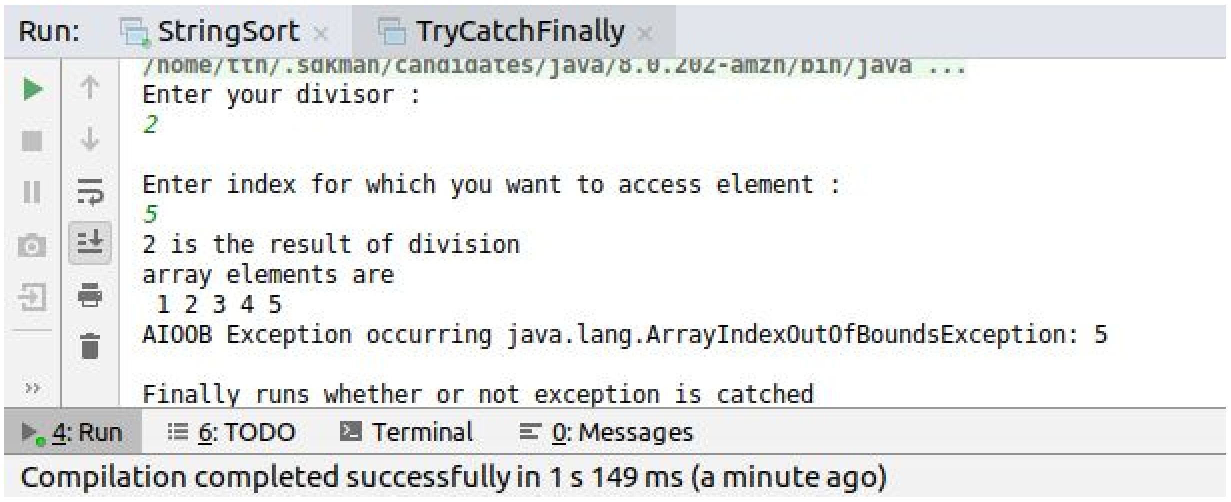
Compilation completed successfully in 1 s 182 ms (a minute ago)

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

```
import java.util.Scanner;

public class TryCatchFinally {


   public static void main(String[] args) {
       int i=5;
```

```
      Scanner in=new Scanner(System.in);
      System.out.println("Enter your divisor : ");
      int j=in.nextInt();

      System.out.println("\nEnter index for which you
want to access element : ");
      int k=in.nextInt();
       int[] a ={1,2,3,4,5};
      try{
         System.out.println(i/j +" is the result of
division");
         System.out.println("array elements are ");
         for(int k1=0; k1<a.length;k1++)
            System.out.print(" "+a[k1]);


         System.out.println("try to access elements of
array "+a[k]);
      }
      catch (ArithmeticException e)
      {
         System.out.println("Arithmetic Exception
occurring "+e);
      }

      catch(ArrayIndexOutOfBoundsException e)
```

```
        {
            System.out.println("\nAIOOB Exception
occurring "+e);

        }

        finally {
            System.out.println("\nFinally runs whether or
not exception is catched");
        }
    }



}
```



```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Enter your divisor :
2

Enter index for which you want to access element :
5
2 is the result of division
array elements are
 1 2 3 4 5
AIOOB Exception occurring java.lang.ArrayIndexOutOfBoundsException: 5

Finally runs whether or not exception is catched
```

Compilation completed successfully in 1 s 149 ms (a minute ago)

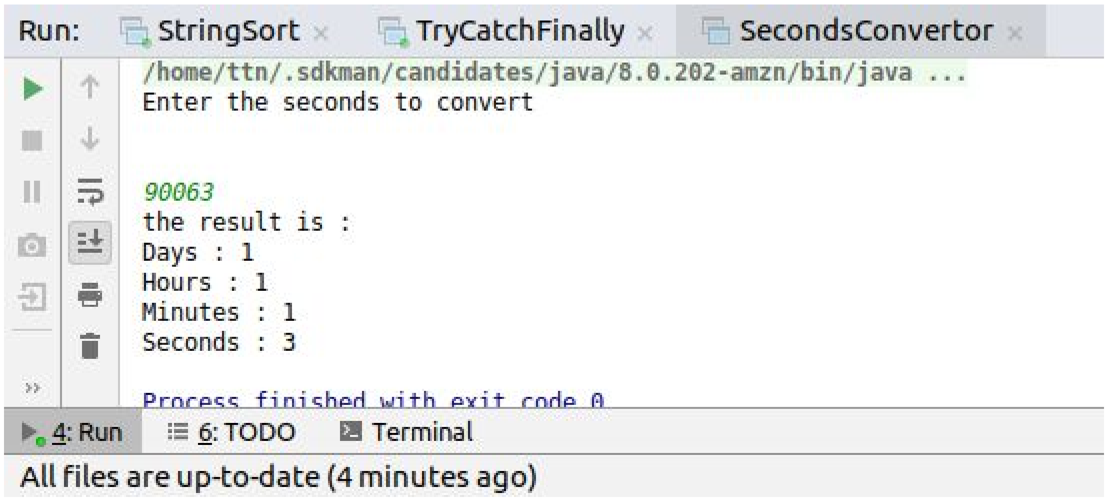
7. WAP to convert seconds into days, hours, minutes and seconds.

```
import java.util.Scanner;

public class SecondsConvertor
{
  int seconds;
  int days;
  int hours;
  int minutes;

  SecondsConvertor(int s)
  {
     seconds = s;
  }

  void convertSeconds()
  {
     days = seconds / 86400;

     int remainingSeconds = seconds % 86400;
    // System.out.println("hours" +
remainingSeconds);

     hours = remainingSeconds / 3600;

     remainingSeconds = remainingSeconds % 3600;
```

```
        //System.out.println("minutes" +
remainingSeconds);

        minutes = remainingSeconds / 60;


        remainingSeconds = remainingSeconds % 60;

        //System.out.println("seconds" +
remainingSeconds);
        seconds = remainingSeconds;
    }

    public static void main(String[] args)
    {
       final int seconds_input;
        System.out.println("Enter the seconds to
convert\n\n");

        try {
            Scanner in = new Scanner(System.in);
             seconds_input= in.nextInt();



             if(seconds_input<0)
```

```
          {
              System.out.println("enter positive values  or 
0 ");
              System.out.exit(0);

          }

         SecondsConvertor converter = new 
SecondsConvertor(seconds_input);
         converter.convertSeconds();

         System.out.println("the result is : ");
         System.out.println("Days : "+converter.days);
         System.out.println("Hours : "+converter.hours);
         System.out.println("Minutes : 
"+converter.minutes);
         System.out.println("Seconds : 
"+converter.seconds);

      }
      catch(Exception e)
      {
         System.out.println("not an int : "+e);
      }

   }
```

```
}
```



Run: StringSort × TryCatchFinally × SecondsConvertor ×

```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Enter the seconds to convert


90063
the result is :
Days : 1
Hours : 1
Minutes : 1
Seconds : 3

Process finished with exit code 0
```

4: Run 6: TODO Terminal

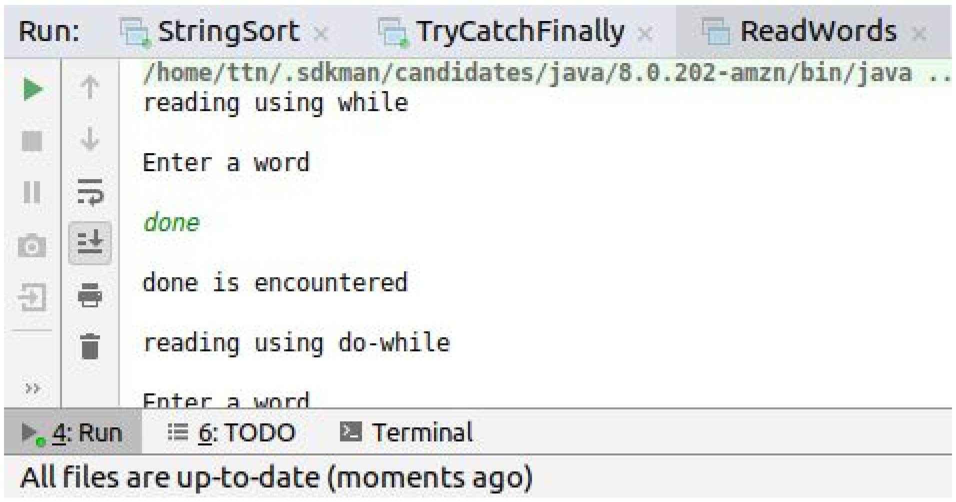
All files are up-to-date (4 minutes ago)

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

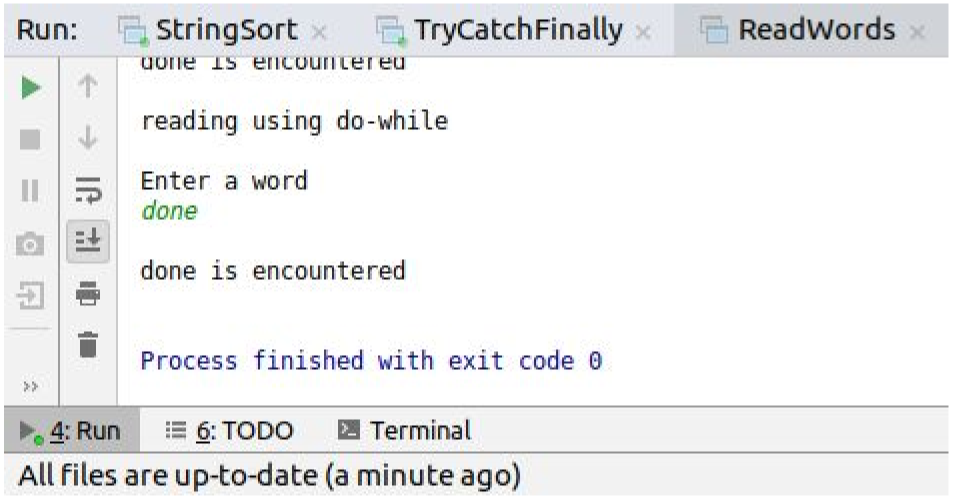
a)while statement
b)do-while statement

```
import java.util.Scanner;

public class ReadWords
{
  public static void main(String[] args)
  {
    System.out.println("reading using while\n");
    Scanner keyboard = new Scanner(System.in);
    System.out.println("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);
        }

      System.out.println("\nEnter a word");
      word = keyboard.next();
    }
    System.out.println("\ndone is encountered\n");
    System.out.println("reading using do-while\n");


    System.out.println("Enter a word");
    word = keyboard.next();
    do
    {
      if(word.equals("done") )
      {
        continue;
      }

      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);
      }
```

```
            System.out.println("\nEnter a word");
            word = keyboard.next();
        }while(!word.equals("done"));

        System.out.println("\ndone is encountered\n");



    }

}
```



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

Enter a word

done

done is encountered

reading using do-while

Enter a word
```



```
done is encountered

reading using do-while

Enter a word
done

done is encountered


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.**

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



abstract class Chair
{
  Chair()
  {
     System.out.println("\n\n\nThis is a chair\n");
  }
}



abstract class Table
{
  Table()
  {
     System.out.println("\n\n\nThis is a table\n");
  }
}



class WoodenChair extends Chair implements
Furniture
{
  static String chairType = "wooden";

  WoodenChair()
  {
     super();
```

```
      System.out.println("\nThis is a " + chairType + "
chair");
   }

   @Override
   public void fireTest()
   {
      System.out.println("Wooden chairs have low
resistance to fire");
   }

   @Override
   public void stressTest()
   {
      System.out.println("Wooden chairs have
moderate resistance to stress");
   }
}
```

```
class MetallicChair extends Chair implements
Furniture
```

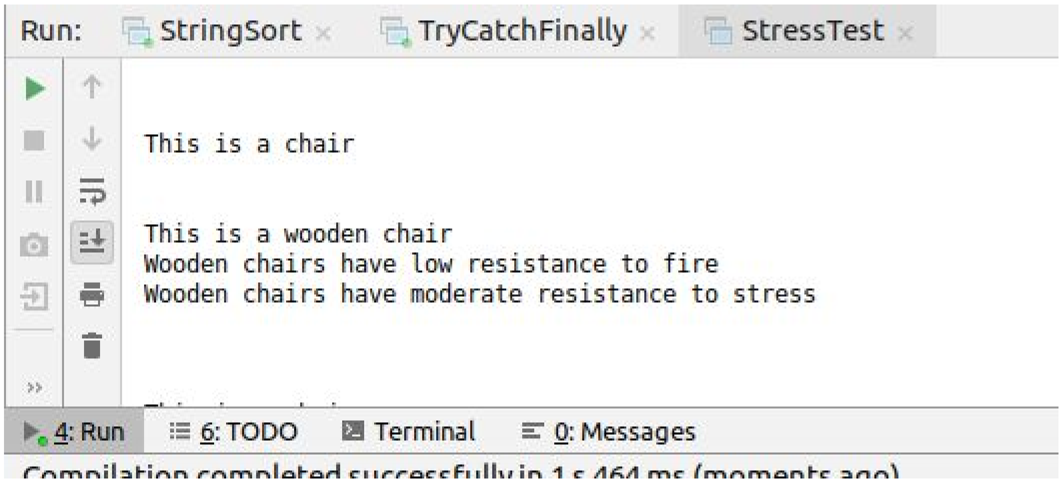
```
{
   static String chairType = "metallic";

   MetallicChair()
   {
      System.out.println("\nThis is a " + chairType + "
chair");
   }

   @Override
   public void fireTest()
   {
      System.out.println("Metallic chairs have high
resistance to fire");
   }

   @Override
   public void stressTest()
   {
      System.out.println("Metallic chairs have high
resistance to stress");
   }
}
```

```
class WoodenTable extends Table implements
Furniture
{
   static String tableType = "wooden";

   WoodenTable()
   {
       super();
       System.out.println("\nThis is a " + tableType + "
table");
   }

   @Override
   public void fireTest()
   {
       System.out.println("Wooden tables have low
resistance to fire");
   }

   @Override
   public void stressTest()
```

```
  {
     System.out.println("Wooden tables have moderate
resistance to stress");
  }
}

class MetallicTable extends Table implements
Furniture
{
  static String tableType = "Metallic";

  MetallicTable()
  {
     System.out.println("\nThis is a " + tableType + "
table");
  }

  @Override
  public void fireTest()
  {
     System.out.println("Metallic tables have high
resistance to fire");
  }

  @Override
  public void stressTest()
  {
     System.out.println("Metallic tables have high
resistance to stress");
  }
}
```

```
public class StressTest
{
  public static void main(String[] args)
  {
     WoodenChair woodenChair = new WoodenChair();
     woodenChair.fireTest();
     woodenChair.stressTest();
     MetallicChair metallicChair = new MetallicChair();
     metallicChair.fireTest();
     metallicChair.stressTest();
     WoodenTable woodenTable = new
WoodenTable();
     woodenTable.fireTest();
     woodenTable.stressTest();
     MetallicTable metallicTable = new MetallicTable();
     metallicTable.fireTest();
```

```
        metallicTable.stressTest();
    }
}
```



```
This is a chair


This is a wooden chair
Wooden chairs have low resistance to fire
Wooden chairs have moderate resistance to stress
```

Compilation completed successfully in 1 s 464 ms (moments ago)

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

```
import java.util.LinkedList;
import java.util.Queue;






class Barista
{
   String baristaId;


   public Barista(String baristaId) {
       this.baristaId = baristaId;
   }

   void getNextOrder()
   {
       System.out.println("getting the next orde from the
order queue");
   }

   void prepareCoffee(){
       System.out.println("your coffee is being
prepared");
   }
   void generateToken(){
       System.out.println("you have been assigned a 
token...your coffee will arrive soon!!");
   }

   void serveCoffee(){
       System.out.println("enjoy your coffee!!");
   }

}
```

```
class Cashier {
  String cashierId;

  public Cashier(String cashierId) {
     this.cashierId = cashierId;
  }

  void createNewOrder(){
//       Creating new Order
//       new Order("id", "item", 45)
     return;
  }

  void giveToken(){
//       assign tokens to espective orders
  }

}
```

```
class Customer {
   String customerId;
   String tokenNumber;

   public Customer(String customerId, String
tokenNumber) {
       this.customerId = customerId;
       this.tokenNumber = tokenNumber;
   }


   void placeOrder(Cashier cashier) {
//        tokenNumber = cashier.createNewOrder();
//        place order
   }

   void enjoyDrink() {
       System.out.println("enjoy your coffee...Thank you
for visiting us!!");
   }

}
```

```
  class Order {
     String orderId;
     String itemName;
     double itemPrice;
     int orderCount;

     public Order(String orderId, String itemName,
double itemPrice) {
        this.orderId = orderId;
        this.itemName = itemName;
        this.itemPrice = itemPrice;
        addOrderToList(orderCount++);
     }

     void addOrderToList(int orderCount)
     {
//       some implemntation
     }
  }


  public class CafeCoffe {
```

```
  //    private static Customer customerQueue;
  private Barista barista;
  private Cashier cashier;
  Queue<Customer> customerQueue;
  Queue<Order> orderQueue = new LinkedList<>();

  public CafeCoffe() {
     System.out.println("Welcome to Cafe Coffee...!!");
     this.orderQueue = new LinkedList<>();
  }


  public void setCustomerQueue(Queue<Customer>
customerQueue) {
     this.customerQueue = customerQueue;
  }

  public void setOrderQueue(Queue<Order>
orderQueue) {
     this.orderQueue = orderQueue;
  }


  public void setBarista(Barista barista) {
     this.barista = barista;
```

```
    }

    public void setCashier(Cashier cashier) {
        this.cashier = cashier;
    }

    public void startServingCustomers(Cashier cashier)
    {
        for(Customer c: this.customerQueue){
            c.placeOrder(cashier);
        }
    }

    public static void main(String[] args) {
        CafeCoffe cafeCoffe =  new CafeCoffe();
        Barista barista = new Barista("b1");
        Cashier cashier = new Cashier("c1");
        cafeCoffe.setBarista(barista);
        cafeCoffe.setCashier(cashier);
//        create customers
        Customer customer1 = new Customer("cst1",
"t1");
        Customer customer2 = new Customer("cst2",
"t2");
        Customer customer3 = new Customer("cst3",
"t3");
```

```
		Customer customer4 = new Customer("cst4",
"t4");
//		adding customers to the custome Queue
		cafeCoffe.customerQueue.add(customer1);
		cafeCoffe.customerQueue.add(customer2);
		cafeCoffe.customerQueue.add(customer3);
		cafeCoffe.customerQueue.add(customer4);
		Queue<Customer> customerQueue = new
LinkedList<>();
		cafeCoffe.setCustomerQueue(customerQueue);
//		orders
		Order order1 = new Order("O1", "abc", 100);
		Order order2 = new Order("O2", "xyz", 500);
		Order order3 = new Order("O3", "efg", 700);
		Order order4 = new Order("O4", "rst", 200);
//		 adding orders to the order queue
		cafeCoffe.orderQueue.add(order1);
		cafeCoffe.orderQueue.add(order2);
		cafeCoffe.orderQueue.add(order3);
		cafeCoffe.orderQueue.add(order4);
		Queue<Order>  orderQueue = new LinkedList<>();
		cafeCoffe.setOrderQueue(orderQueue);

//		Begin opeations
		cafeCoffe.startServingCustomers(cashier);
```

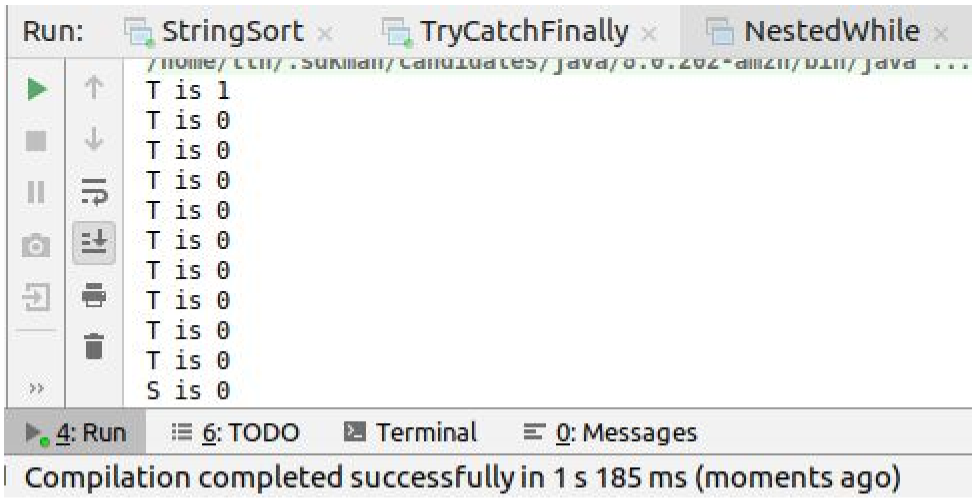
```
    }
}
```

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);
```

```
public class NestedWhile {
```

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



Run: StringSort × TryCatchFinally × NestedWhile ×

```
/home/ctn/.sdkman/candidates/java/8.0.202-amzn/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
```

4: Run 6: TODO Terminal 0: Messages

Compilation completed successfully in 1 s 185 ms (moments ago)

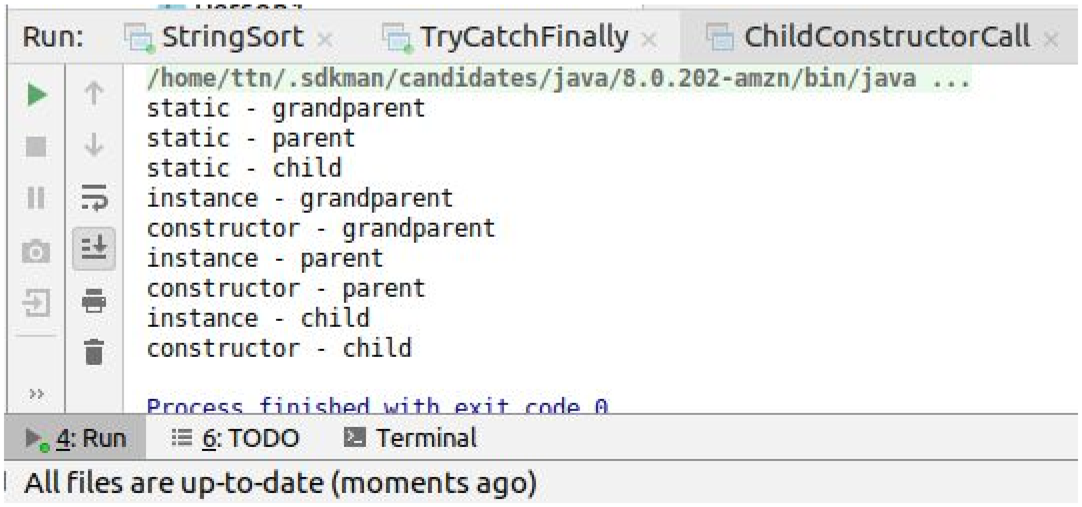
12.What will be the output on new Child(); ?

```
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");
        }
    }


class Parent extends Grandparent
{
//instance block of Parent
```

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

    public Parent() //constructor of Parent
    {
        System.out.println("constructor - parent");
    }

    static //static block of Parent
    {
        System.out.println("static - parent");
    }
}

class Grandparent
{

    static //static block of Grandparent
    {
        System.out.println("static - grandparent");
    }




    //instance block of Grandparent
    {
       System.out.println("instance - grandparent");
    }

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

class Child extends Parent
{

    public Child() //constructor of Child
    {
       System.out.println("constructor - child");
    }

    static //static block of Child
    {
       System.out.println("static - child");
    }


    //instance block of Child
    {
```

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

```java
public class ChildConstructorCall
{
    public static void main(String[] args) {
        new Child();
    }
}
```



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

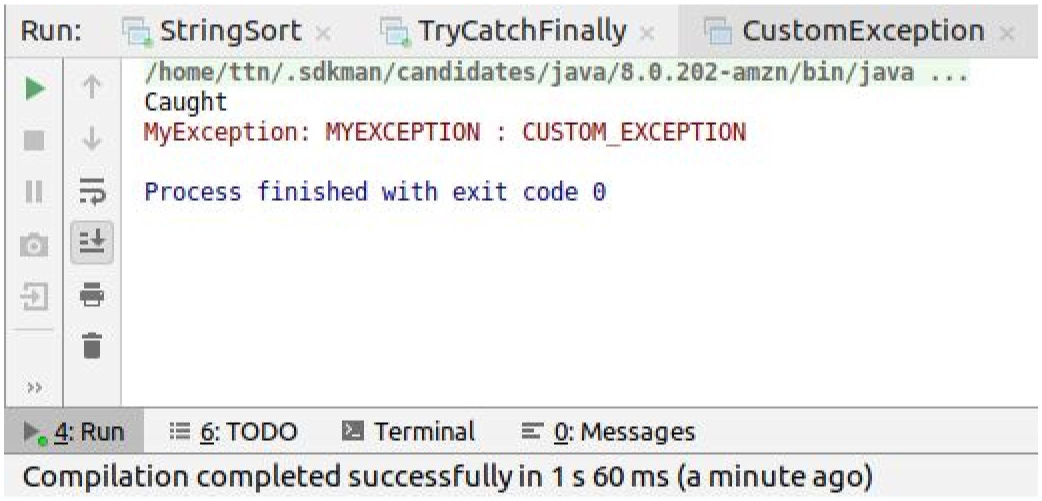
All files are up-to-date (moments ago)

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

```
class MyException extends Exception{
  MyException(String s)
  {
     super("MYEXCEPTION : "+s);
  }

  public  Throwable fillInStackTrace() {
     return this;
  }




}
```

```
class CustomExceptionTest
{
   void testingException() throws MyException
   {
      throw new MyException("CUSTOM_EXCEPTION");
   }
}







public class CustomException {


   public static void main(String args[]){

      try { caller();}
      catch(MyException e)
      {
         System.out.println("Caught");


     //    e.fillInStackTrace();
         e.printStackTrace();
```

```
        }

    }



    public static void caller() throws MyException
    {
        CustomExceptionTest exception1=new
CustomExceptionTest();



            exception1.testingException();// Throw an
object of user defined exception



    }


}
```



```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Caught
MyException: MYEXCEPTION : CUSTOM_EXCEPTION

Process finished with exit code 0
```

Compilation completed successfully in 1 s 60 ms (a minute ago)