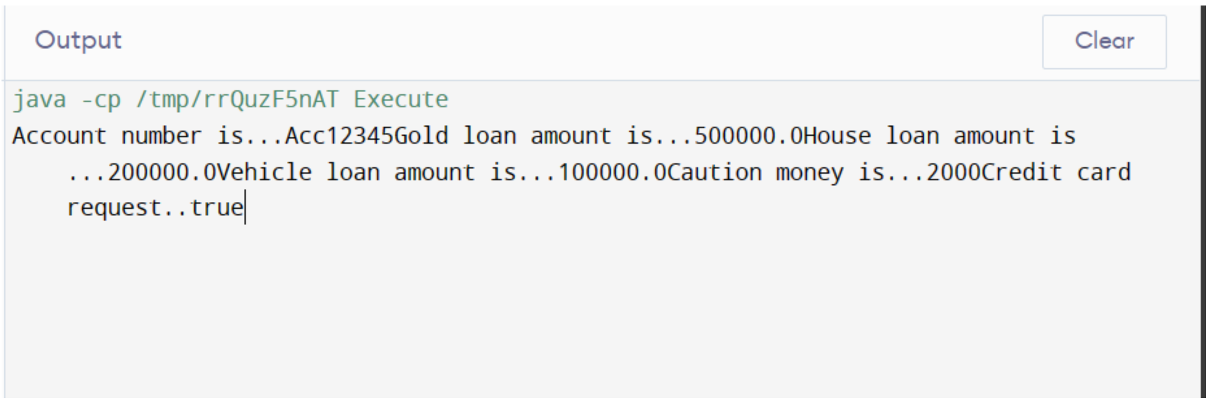
DAY 4

1] interface

```
 interface IBankNew{
   boolean applyforCreditCard(Customer customer);
}

interface IBank extends IBankNew{
   int CAUTION_MONEY = 2000;
   String createAccount(Customer customer);
   double issueVehicleLoan(String vehicleType, Customer customer);
   double issueHouseLoan(Customer customer);
   double issueGoldLoan(Customer customer);
}

class Customer {
   private String name;
   private String customerId;

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name=name;
   }
   public String getCustomerId() {
      return customerId;
   }
   public void setCustomerId(String customerId) {
      this.customerId= customerId;
   }
}

class MumbaiBranch implements IBank {
   public String createAccount(Customer customer){
      return "Acc12345";
   }
   public double issueVehicleLoan(String vehicleType,Customer customer){
      if(vehicleType.equals("bike")) {
         return 100000;
      }
      else {
         return 500000;
      }
   }
   public double issueHouseLoan(Customer customer){
      return 200000;
```

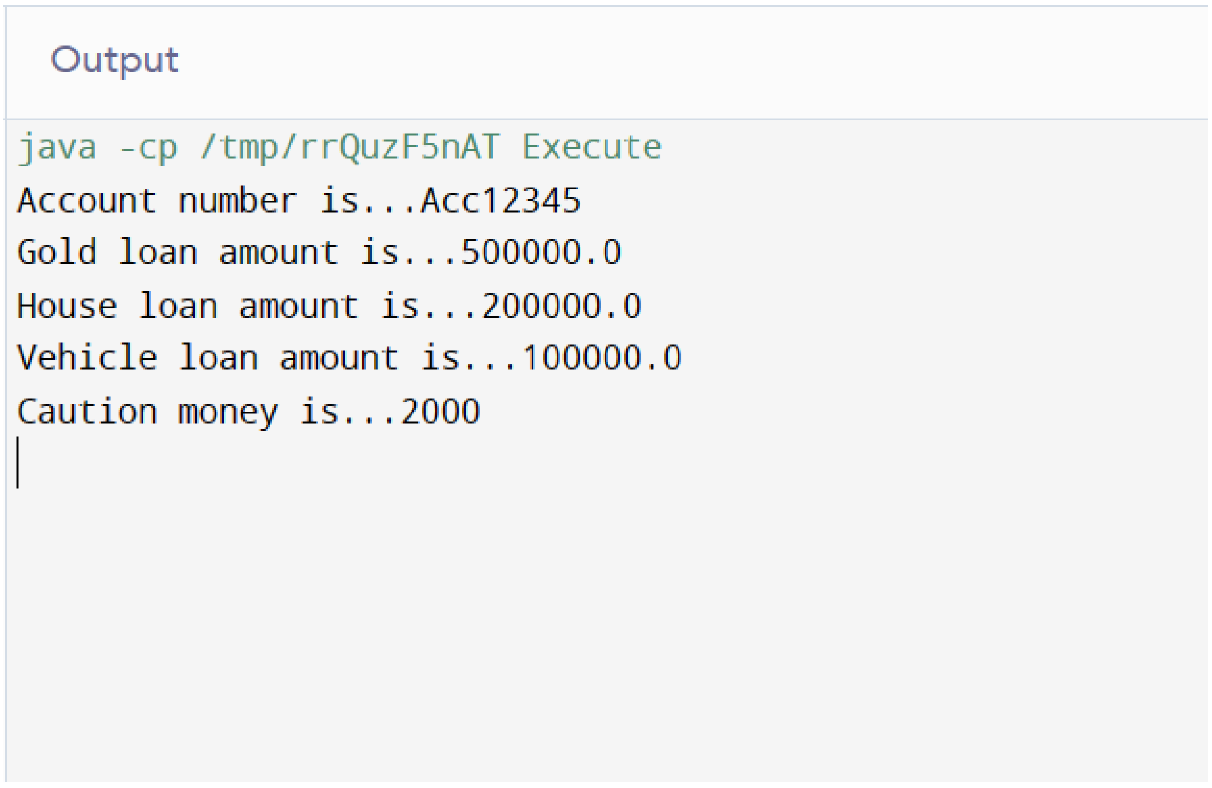
```
    }
    public double issueGoldLoan(Customer customer){
        return 500000;
    }
    public boolean applyforCreditCard(Customer customer){
        return true;
    }
}

class Execute{
    public static void main(String[] args){
        IBank bank=new MumbaiBranch();
        Customer customer = new Customer();
        customer.setCustomerId("cust1001");
        customer.setName("Ajay");
        String accountNumber = bank.createAccount(customer);
        System.out.println("Account number is..." +accountNumber);
        double gloan = bank.issueGoldLoan(customer);
        System.out.println("Gold loan amount is..." +gloan);
        double hloan = bank.issueHouseLoan(customer);
        System.out.println("House loan amount is..." +hloan);
        double vloan = bank.issueVehicleLoan("bike", customer);
        System.out.println("Vehicle loan amount is..." +vloan);
        System.out.println("Caution money is..." +IBank.CAUTION_MONEY);
        IBankNew bank1 = new MumbaiBranch();
        boolean credit = bank1.applyforCreditCard(customer);
        System.out.println("Credit card request.." + credit);
    }
}
```



```
java -cp /tmp/rrQuzF5nAT Execute
Account number is...Acc12345Gold loan amount is...500000.0House loan amount is
    ...200000.0Vehicle loan amount is...100000.0Caution money is...2000Credit card
    request..true
```

2] interface

```
interface IBank {
    int CAUTION_MONEY = 2000;
    String createAccount(Customer customer);
    double issueVehicleLoan(String vehicleType, Customer customer);
    double issueHouseLoan(Customer customer);
    double issueGoldLoan(Customer customer);
}
class Customer {
    private String name;
    private String customerId;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name=name;
    }
    public String getCustomerId() {
        return customerId;
    }
    public void setCustomerId(String customerId) {
        this.customerId= customerId;
    }
}
class MumbaiBranch implements IBank {
    public String createAccount(Customer customer){
        return "Acc12345";
    }
    public double issueVehicleLoan(String vehicleType,Customer customer){
        if(vehicleType.equals("bike")) {
            return 100000;
        }
        else {
            return 500000;
        }
    }
    public double issueHouseLoan(Customer customer){
        return 200000;
    }
    public double issueGoldLoan(Customer customer){
        return 500000;
    }
}

class Execute{
    public static void main(String[] args){
        IBank bank=new MumbaiBranch();
        Customer customer = new Customer();
        customer.setCustomerId("cust1001");
        customer.setName("Ajay");
        String accountNumber = bank.createAccount(customer);
        System.out.println("Account number is..." +accountNumber);
        double gloan = bank.issueGoldLoan(customer);
        System.out.println("Gold loan amount is..." +gloan);
        double hloan = bank.issueHouseLoan(customer);
        System.out.println("House loan amount is..." +hloan);
        double vloan = bank.issueVehicleLoan("bike", customer);
        System.out.println("Vehicle loan amount is..." +vloan);
        System.out.println("Caution money is..." +IBank.CAUTION_MONEY);
    }
}
```



Output

```
java -cp /tmp/rrQuzF5nAT Execute
Account number is...Acc12345
Gold loan amount is...500000.0
House loan amount is...200000.0
Vehicle loan amount is...100000.0
Caution money is...2000
```

3] access modifiers

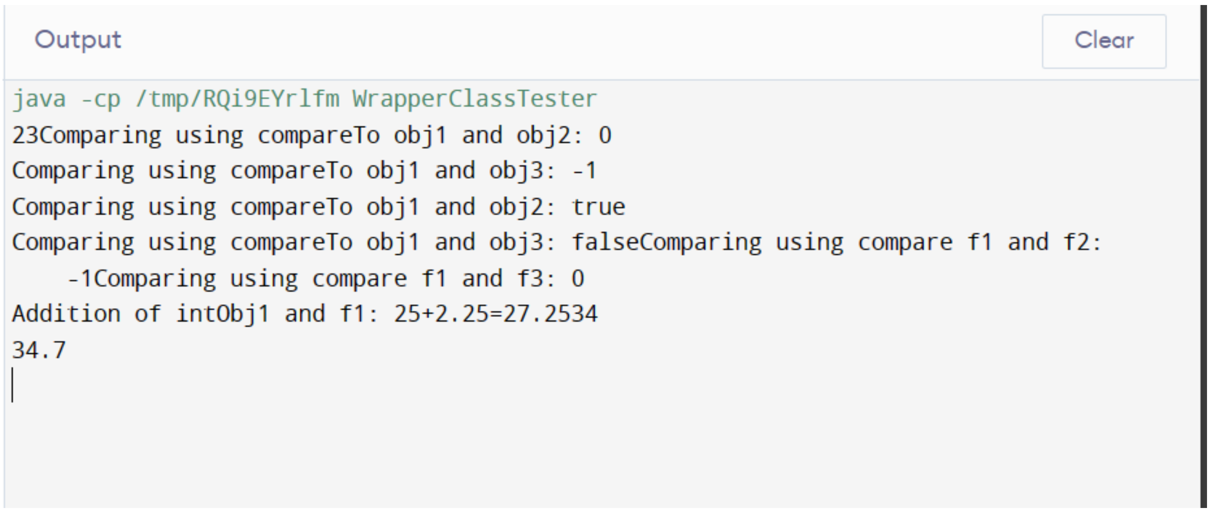
```
class Person{
    private int salary = 5000;
    public String name = "Jack";
    protected int age = 24;
    String email = "jack@samurai.com";

    public void display(){
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Email: " + email);
        System.out.println("Salary: " + salary);
    }
}
```

```
class Employee extends Person {
    public void display(){
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Email: " + email);
    }
}

class Customer {
    public void display(){
        Person p = new Person();
        System.out.println("Name: " + p.name);
        System.out.println("Age: " + p.age);
        System.out.println("Email: " + p.email);
    }
}

class Execute{
    public static void main (String[] args) {
        Person p = new Person();
        Employee e = new Employee();
        Customer c =  new Customer();
        System.out.println("******************************");
        System.out.println("Person Class display method.");
        System.out.println("******************************");
        p.display();
        System.out.println("******************************");
        System.out.println("Employee Class display method.");
        System.out.println("******************************");
        e.display();
        System.out.println("******************************");
        System.out.println("Customer Class display method.");
        System.out.println("******************************");
        c.display();
    }
}
```

4] wrapper

```
 class WrapperClassTester {

    public static void main(String[] args) {

    int i = 45;//primitive data int
```

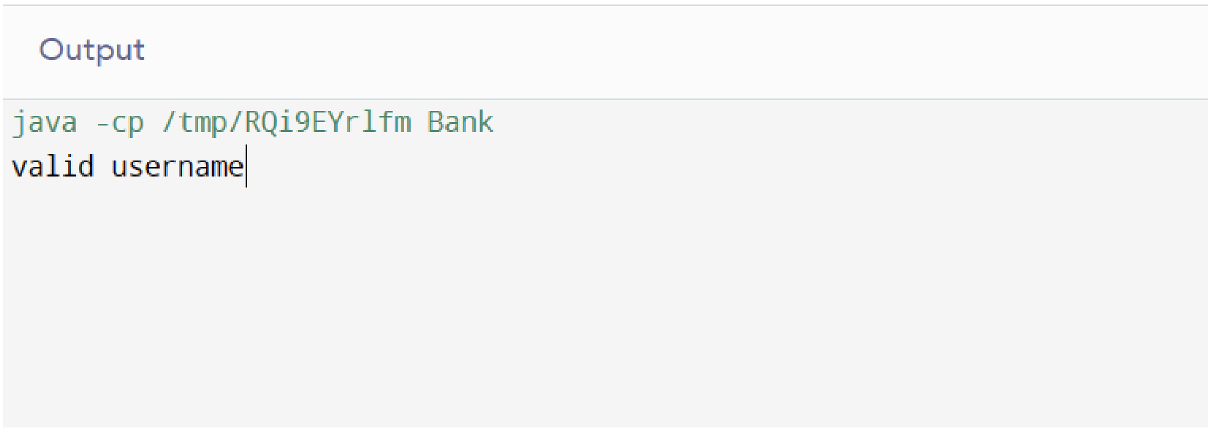
```
   Integer integer = new Integer(i);// Integer wrapper class instantiation
   int i2 = integer.intValue();// unwrapping primitive data int from wrapper object
   Integer integer2 = new Integer("23");

   // all wrapper class except Character can take String in argument
   System.out.println(integer2);
   Integer intObj1 = new Integer(25);
   Integer intObj2 = new Integer("25");
   Integer intObj3 = new Integer(35);

   //compareTo demo
   System.out.println("Comparing using compareTo obj1 and obj2: " +
intObj1.compareTo(intObj2));
   System.out.println("Comparing using compareTo obj1 and obj3: " +
intObj1.compareTo(intObj3));

   // Equals demo
   System.out.println("Comparing using compareTo obj1 and obj2: " +
intObj1.equals(intObj2));
   System.out.println("Comparing using compareTo obj1 and obj3: " +
intObj1.equals(intObj3));
   Float f1 = new Float("2.25f");
   Float f2 = new Float("20.43f");
   Float f3 = new Float(2.25f);
   System.out.println("Comparing using compare f1 and f2: " + Float.compare(f1,f2));
   System.out.println("Comparing using compare f1 and f3: " + Float.compare(f1,f3));

   // Addition of Integer with Float
   Float f = intObj1.floatValue() + f1;
   System.out.println("Addition of intObj1 and f1: "+ intObj1 + "+" + f1 + "=" + f);
   int x = Integer.parseInt("34");
   System.out.println(x);
   double y = Double.parseDouble("34.7");
   System.out.println(y);
   }
 }
```



```
java -cp /tmp/RQi9EYrlfm WrapperClassTester
23Comparing using compareTo obj1 and obj2: 0
Comparing using compareTo obj1 and obj3: -1
Comparing using compareTo obj1 and obj2: true
Comparing using compareTo obj1 and obj3: falseComparing using compare f1 and f2:
    -1Comparing using compare f1 and f3: 0
Addition of intObj1 and f1: 25+2.25=27.2534
34.7
```

5] string class

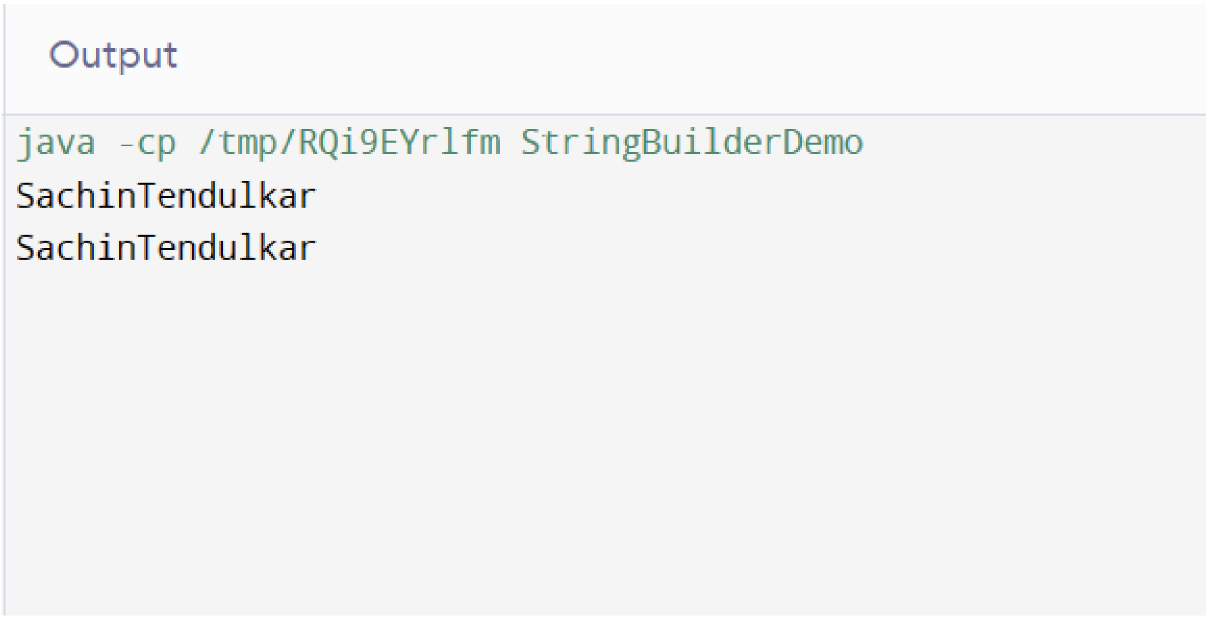
```
class Bank{
  public static void main(String[] args){
    String username = "Tendulkar";
    int size = username.length();
    if(size > 8 && size <15){
      char arr[]=username.toCharArray();
      int count=0;
      for(char c:arr){
        if(Character.isLetter(c)){
          ++count;
        }
      }
      if(count == size){
        System.out.println("valid username");
      }
    }
  }
}
```



```
java -cp /tmp/RQi9EYrlfm Bank
valid username
```

6]
String builder

```
class StringBuilderDemo{

	public static void main(String[] args){

		String firstName="Sachin";
		String lastName="Tendulkar";
		String fullName=firstName+lastName;
		//'+'operator concatenates the string but creates a new object in the heap
memory as sting is immutable
		System.out.println(fullName);
		StringBuilder sb=new StringBuilder(firstName);
		String fName=sb.append(lastName).toString();//toString method converts
StringBuilder to String
		//StringBuilder is mutable, it appends to a single object
		System.out.println(fName);

	}
}
```



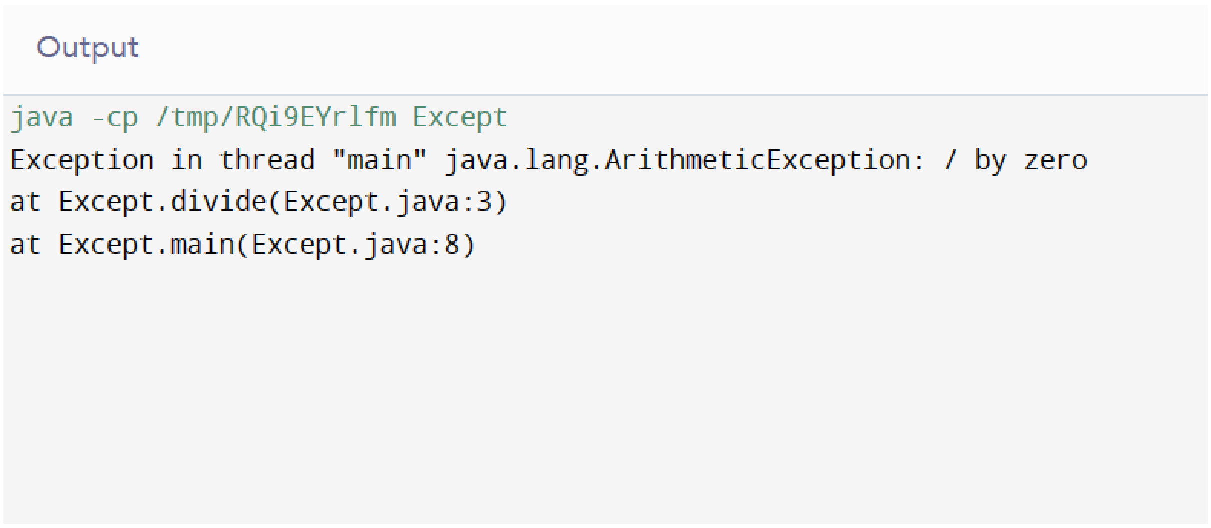
Output

```
java -cp /tmp/RQi9EYrlfm StringBuilderDemo
SachinTendulkar
SachinTendulkar
```

7] exception demonstration

```
class Except {
	public static void divide(int x, int y) {
		int z = x / y;
		System.out.println(z);
	}

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

```
                    divide(10, 0);
        }
}
```

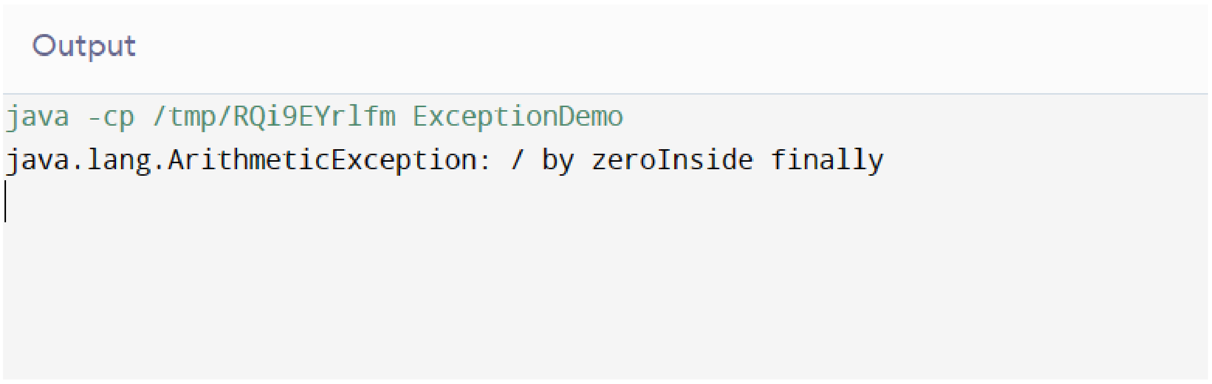


Output

```
java -cp /tmp/RQi9EYrlfm Except
Exception in thread "main" java.lang.ArithmeticException: / by zero
at Except.divide(Except.java:3)
at Except.main(Except.java:8)
```

8] try - catch - finally

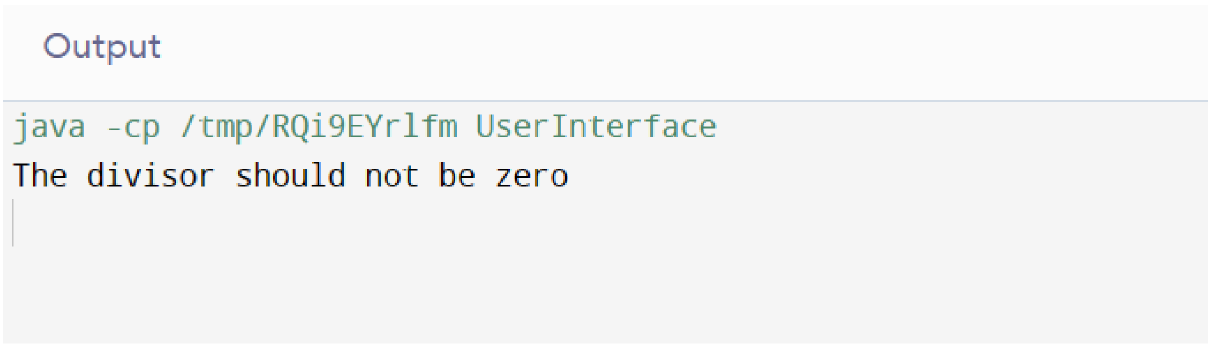
```
class ExceptionDemo {

        public static int divide(int a,int b) {
                return a/b;
        }

        public static void main(String[] args) {
                try {
                        divide(9,0);
                } catch (ArithmeticException exception) {
                        System.out.println(exception);
                        //exception.printStackTrace();
                        //System.out.println(exception.getMessage());
                        //System.out.println(exception.toString());
                }
      finally  {
        System.out.println("Inside finally");
     }
        }
}
```



```
java -cp /tmp/RQi9EYrlfm ExceptionDemo
java.lang.ArithmeticException: / by zeroInside finally
```

9] throw

```
class UserInterface {
        public static void divide(int x, int y) {
                try {
                        if (y == 0)
                                throw new Exception("The divisor should not be zero");
                        int z = x / y;
                        System.out.println(z);
                } catch (Exception e) {
                        System.out.println(e.getMessage());
                }
        }

        public static void main(String[] args) {
                UserInterface.divide(10, 0);
        }
}
```

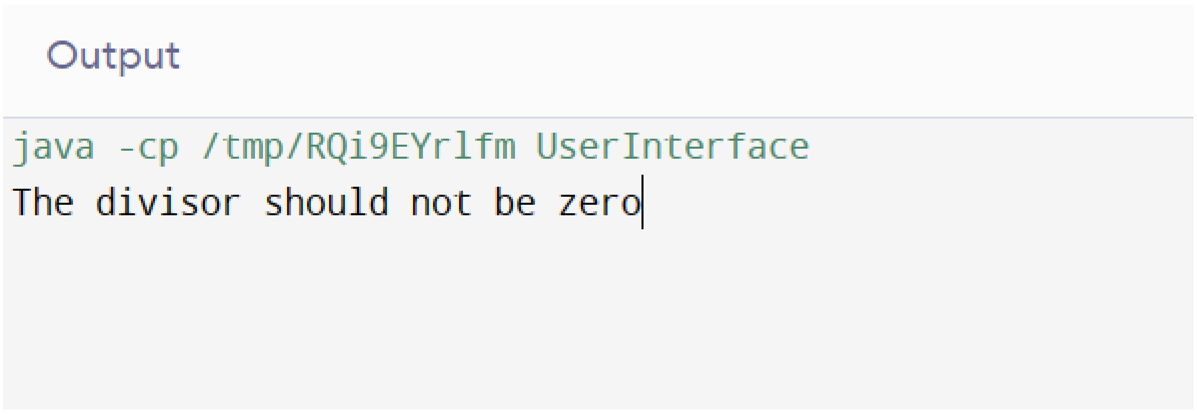


```
java -cp /tmp/RQi9EYrlfm UserInterface
The divisor should not be zero
```

10] Throw clause

```
class UserInterface{
   public static void divide(int x, int y) throws Exception {
     if(y == 0)
         throw new Exception("The divisor should not be zero");
     int z = x/y;
```

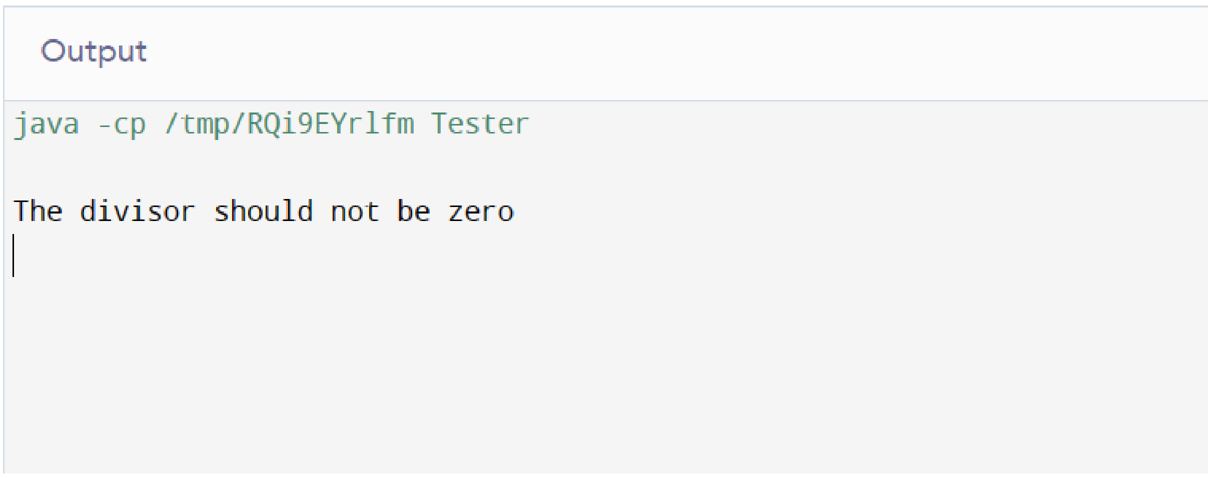
```
        System.out.println(z);
}
public static void main(String[] args) {
        try {
            divide(10, 0);
        }
        catch(Exception e) {
            System.out.println(e.getMessage());
        }
}
}
```


Output

```
java -cp /tmp/RQi9EYrlfm UserInterface
The divisor should not be zero
```

11] user defined Exceptions

```
class MyDivException extends Exception
{
    public MyDivException(String message) {
        super(message);
    }
}

class Tester
{
    public static void divide(int x, int y) throws MyDivException {
        if(y == 0)
           throw new MyDivException("The divisor should not be zero");
        int z = x/y;
            System.out.println(z);
    }

    public static void main(String[] args)
    {
        try
        {
        divide(6,0);
        }catch(MyDivException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

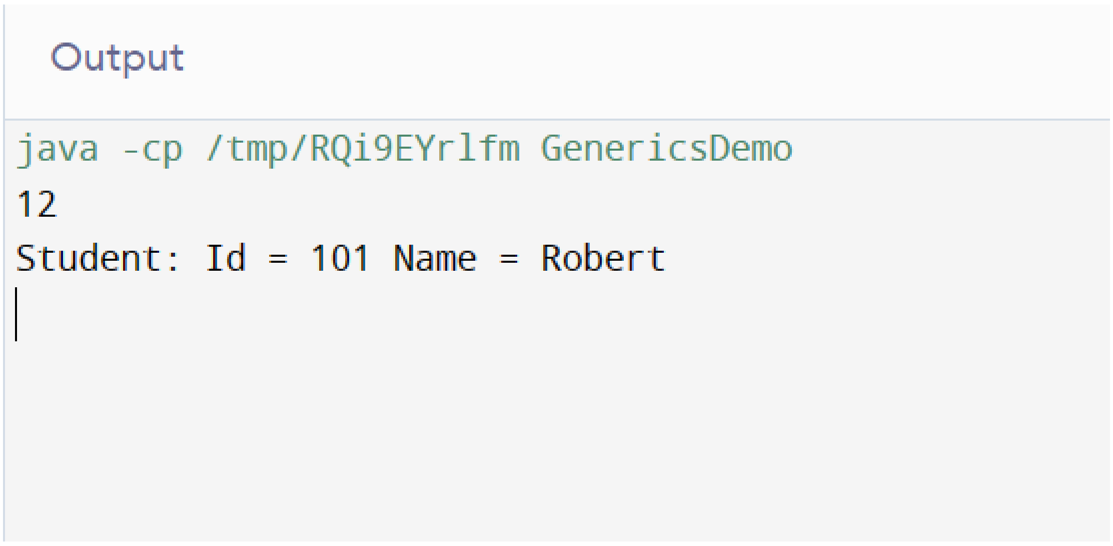


Output

```
java -cp /tmp/RQi9EYrlfm Tester

The divisor should not be zero
```

13] Generic

```
class Record<E> {
    private E record;
    public void display(E item) {
        System.out.println(item);
    }
}

class Student {
    private int studentId;
    private String studentName;

    public Student(int studentId,String studentName)
    {
        this.studentId=studentId;
        this.studentName=studentName;
    }
    public String toString()
    {
        return "Student: Id = " + studentId + " Name = " + studentName;
    }
}

class GenericsDemo {
    public static void main(String[] args)
    {
        Student s1 = new Student(101,"Robert");
        Record<Integer> integerRecord = new Record<Integer>(); //integerRecord can be used
to display only integers
        integerRecord.display(12);
        //integerRecord.display(s1); will give an error as we are trying to add a student class
object
        Record<Student> studentRecord = new Record<>();  //studentRecord can be used to
display only Students
        studentRecord.display(s1);
        //studentRecord.display(15); will give an error as we are trying to add an integer
    }
}
```



Output

```
java -cp /tmp/RQi9EYrlfm GenericsDemo
12
Student: Id = 101 Name = Robert
```

Exercise:

1] Interface:

```
// Interface defining constants
interface Constants {
    int TOTAL_MAXIMUM_MARKS = 8000;
    int GRACE_MARKS_INSTITUTE_A = 100;
    int MARKS_FOR_COURSES_INSTITUTE_A = 900;
    int MARKS_FOR_COURSES_INSTITUTE_B = 1000;
}

// Interface defining methods
interface PercentageCalculator {
    double calcPercentage(int marksSecured, int graceMarks, int marksForCourses);
}

// Intern class implementing the interface
class Intern implements PercentageCalculator, Constants {
    private int marksSecured;
    private int graceMarks;

    // Constructor
    public Intern(int marksSecured, int graceMarks) {
        this.marksSecured = marksSecured;
        this.graceMarks = graceMarks;
    }

    // Override the interface method
    @Override
    public double calcPercentage(int marksSecured, int graceMarks, int marksForCourses) {
        if (marksForCourses != MARKS_FOR_COURSES_INSTITUTE_A) {
            System.out.println("Please enter the correct marks for Institute A.");
            return -1; // indicating an error
        }

        int totalMarks = this.marksSecured + this.graceMarks;
        return ((double) totalMarks / TOTAL_MAXIMUM_MARKS) * 100;
    }
}

// Trainee class implementing the interface
class Trainee implements PercentageCalculator, Constants {
    private int marksSecured;

    // Constructor
    public Trainee(int marksSecured) {
        this.marksSecured = marksSecured;
    }

    // Override the interface method
    @Override
    public double calcPercentage(int marksSecured, int graceMarks, int marksForCourses) {
        if (marksForCourses != MARKS_FOR_COURSES_INSTITUTE_B) {
            System.out.println("Please enter the correct marks for Institute B.");
            return -1; // indicating an error
        }

        return ((double) this.marksSecured / TOTAL_MAXIMUM_MARKS) * 100;
    }
}

public class Main {
    public static void main(String[] args) {
        // Example usage for Intern
        Intern intern1 = new Intern(5000, 500);
        double internPercentage1 = intern1.calcPercentage(intern1.marksSecured,
intern1.graceMarks, MARKS_FOR_COURSES_INSTITUTE_A);
        if (internPercentage1 != -1) {
            System.out.println("Intern 1: The total aggregate percentage secured is " +
internPercentage1);
        }

        // Example usage for Trainee
        Trainee trainee1 = new Trainee(6000);
        double traineePercentage1 = trainee1.calcPercentage(trainee1.marksSecured, 0,
MARKS_FOR_COURSES_INSTITUTE_B);
        if (traineePercentage1 != -1) {
            System.out.println("Trainee 1: The total aggregate percentage secured is " +
traineePercentage1);
        }
    }
}
```

2]