

Java Practical Questions

Q1. E-commerce Order Management (Classes & Objects):

Create a class Order with attributes orderID, customerName, and orderAmount. Implement methods to place an order, display order details, and apply a discount for orders above 5000.

Ecommerce:

```
package E_commerceOrderManagement;

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

        Order o = new Order();
        o.setOrderID(1);
        o.setCustomerName("Kalyani");
        o.setOrderAmount(50000);

        //Place Order
        o.placeOrder();

        //Display Order
        o.display();

        //Apply Discount
        o.discount();

    }
}
```

Order:

```
package E_commerceOrderManagement;

public class Order {

    private int orderID;
    private String customerName;
    private double orderAmount;

    public int getOrderID() {
        return orderID;
    }

    public void setOrderID(int orderID) {
        this.orderID = orderID;
    }

    public String getCustomerName() {
        return customerName;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public double getOrderAmount() {
        return orderAmount;
    }

    public void setOrderAmount(double orderAmount) {
        this.orderAmount = orderAmount;
    }

    public void display() {
        System.out.println("\nOrder Details:");
        System.out.println("Order ID : " + orderID + " " + "\nCustomer Name : " + customerName + " " + "\nOrder Amount : " + orderAmount);
    }

    public void placeOrder() {
        System.out.println("Order Placed!!!");
    }

}
```

Output:

```
Console ×
<terminated> Ecommerce [Java Application] C:\Users\Javlekar\Downloads\spring-tool-suite
Order Placed!!!

Order Details:
Order ID : 1
Customer Name : Kalyani
Order Amount : 50000.0

Discount:
Amount after 10% discount : 45000.0
```

Q2. Ride-Sharing System (Polymorphism):

Define an interface Vehicle with a method calculateFare(distance).

Implement Car and Bike classes that calculate fare differently (e.g., Car: ₹10/km, Bike: ₹5/km).

Use polymorphism to calculate the fare based on user input.

Vehicle:

```
package Ride_SharingSystem;

public interface Vehicle {

    double calculateFare(double distance);

}
```

Car:

```
package Ride_SharingSystem;

public class Car implements Vehicle{

    @Override
    public double calculateFare(double distance) {
        return 10 * distance;
    }

}
```

Bike:

```
package Ride_SharingSystem;

public class Bike implements Vehicle {

    //Method Override --> runtime polymorphism
    @Override
    public double calculateFare(double distance) {
        return 5 * distance;
    }

}
```

Ride Sharing:

```
package Ride_SharingSystem;

import java.util.Scanner;

public class RideSharing {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Distance you want to travel : ");
        double distance = sc.nextDouble();

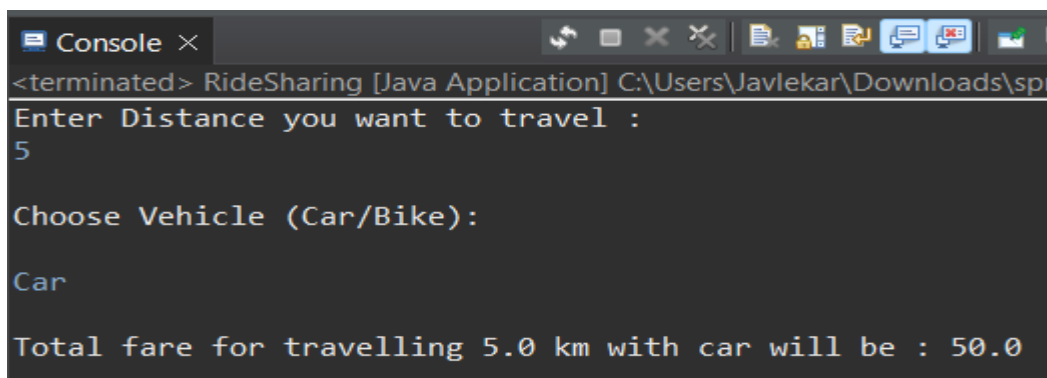
        System.out.println("\nChoose Vehicle (Car/Bike): ");
        String vehicleType = sc.next().toLowerCase();

        Vehicle vehicle = null;

        if (vehicleType.equals("car"))
        {
            vehicle = new Car();
            double fare = vehicle.calculateFare(distance);
            System.out.println("\nTotal fare for travelling " + distance + " km with " + vehicleType + " will be : " + fare);
        }
        else if (vehicleType.equals("bike"))
        {
            vehicle = new Bike();
            double fare = vehicle.calculateFare(distance);
            System.out.println("\nTotal fare for travelling " + distance + " km with " + vehicleType + " will be : " + fare);
        }
        else
        {
            System.out.println("User entered vehicle type not valid!");
        }
    }

}
```

Output:



```
Console x
<terminated> RideSharing [Java Application] C:\Users\Javlekar\Downloads\sp
Enter Distance you want to travel :
5

Choose Vehicle (Car/Bike):
Car

Total fare for travelling 5.0 km with car will be : 50.0
```

```
Console ×
<terminated> RideSharing [Java Application] C:\Users\Javlekar\Downloads\spring
Enter Distance you want to travel :
2

Choose Vehicle (Car/Bike):

Bike

Total fare for travelling 2.0 km with bike will be : 10.0
```

Q3. Smart Home Automation (Abstraction & Interfaces):

Create an abstract class Appliance with methods turnOn() and turnOff(). Implement subclasses Fan and Light, overriding the methods to display appropriate messages.

Appliance:

```
package Smart_HomeAutomation;

public abstract class Appliance {

    abstract void turnON();
    abstract void turnOFF();

}
```

Fan:

```
package Smart_HomeAutomation;

public class Fan extends Appliance{

    @Override
    void turnON() {
        System.out.println("Fan is turning on.");
    }

    @Override
    void turnOFF() {
        System.out.println("Fan is turning off.");
    }

}
```

Light:

```
package Smart_HomeAutomation;

public class Light extends Appliance {

    @Override
    void turnON() {
        System.out.println("Light is turning on.");
    }

    @Override
    void turnOFF() {
        System.out.println("Light is turning on.");
    }

}
```

Smart Home Authentication:

```
package Smart_HomeAutomation;

public class SmartHomeAutomation {

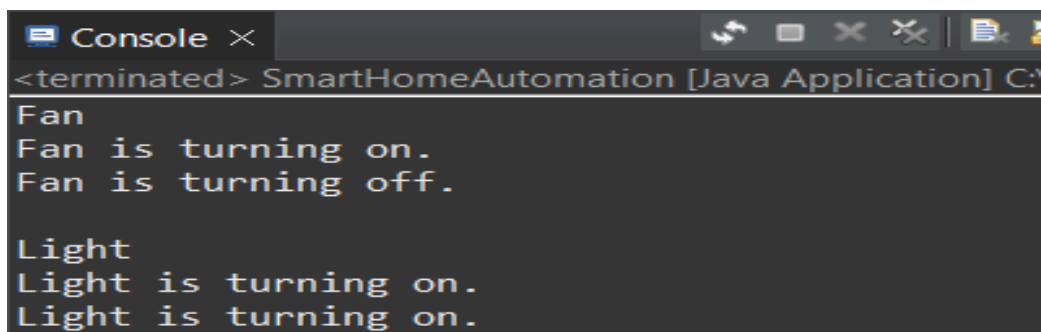
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Fan f = new Fan();
        System.out.println("Fan");
        f.turnON();
        f.turnOFF();

        Light l = new Light();
        System.out.println("\nLight");
        l.turnON();
        l.turnOFF();
    }

}
```

Output:



The screenshot shows a console window titled "Console" with the following output:

```
<terminated> SmartHomeAutomation [Java Application] C:
Fan
Fan is turning on.
Fan is turning off.

Light
Light is turning on.
Light is turning on.
```

Q4. Bank Loan Eligibility Checker:

Ask the user for age, monthly income, and credit score.

Use conditions to determine loan eligibility:

Age < 21 → Not eligible

Income < ₹20,000 → Not eligible

Credit Score < 650 → Low-interest loan

Otherwise → Eligible for a standard loan

```
package Bank_LoanEligibilityChecker;

import java.util.Scanner;

public class BankLoan {

    public static void main(String[] args) {

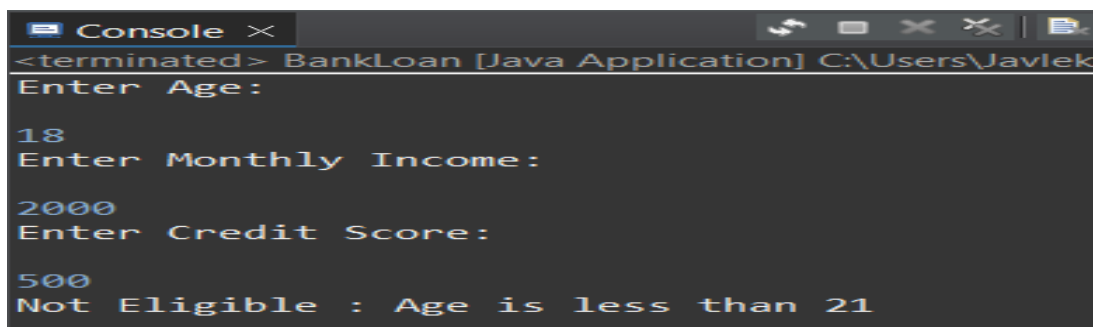
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Age: ");
        int age = sc.nextInt();

        System.out.println("Enter Monthly Income: ");
        double income = sc.nextDouble();

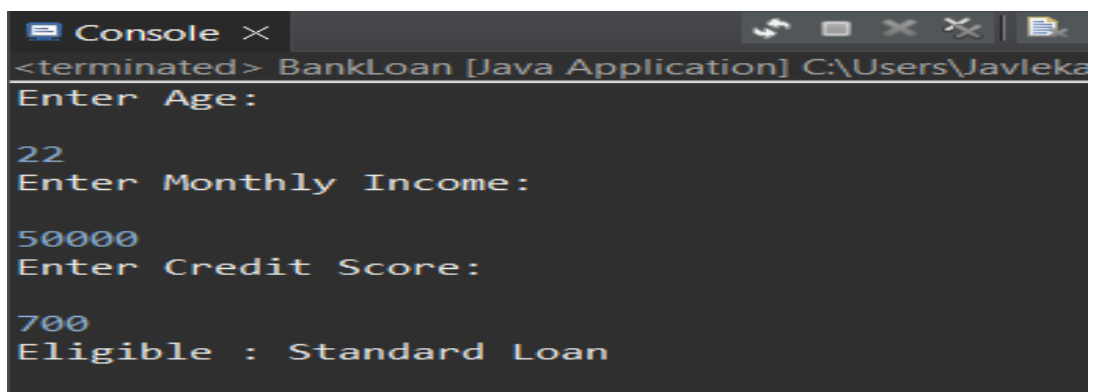
        System.out.println("Enter Credit Score: ");
        double creditScore = sc.nextDouble();

        if (age < 21) {
            System.out.println("Not Eligible : Age is less than 21");
        } else if (income < 20000) {
            System.out.println("Not Eligible : Income is less than 20000");
        } else if (creditScore < 650) {
            System.out.println("Eligible : Low-Interest Loan");
        } else {
            System.out.println("Eligible : Standard Loan");
        }
    }
}
```

Output:



```
Console ×
<terminated> BankLoan [Java Application] C:\Users\Javlek...
Enter Age:
18
Enter Monthly Income:
2000
Enter Credit Score:
500
Not Eligible : Age is less than 21
```



```
Console ×
<terminated> BankLoan [Java Application] C:\Users\Javlek...
Enter Age:
22
Enter Monthly Income:
50000
Enter Credit Score:
700
Eligible : Standard Loan
```

Q5. Online Shopping Cart System (Classes & Objects):

Create a class Product with attributes like productID, name, and price.

Create a class Cart that contains a list of products and methods to add items, remove items, and display the total bill.

Product:

```
package Online_ShoppingCartSystem;

public class Product {

    int productId;
    String name;
    double price;

    Product(int productId, String name, double price) {
        this.productId = productId;
        this.name = name;
        this.price = price;
    }

}
```

Cart:

```
package Online_ShoppingCartSystem;

import java.util.Scanner;

public class Cart {

    Scanner sc = new Scanner(System.in);
    Product[] items = new Product[10];
    int count = 0;

    void addItem(Product p)
    {
        if(count < 10)
        {
            items[count++] = p;

            System.out.println(p.name + " Added");
        }
        else
        {
            System.out.println("Cart Full");
        }
    }

    void removeItems(int productId)
    {
        for(int i = 0; i < count; i++)
        {
            if(items[i].productId == productId)
            {
                items[i] = items[--count];
                items[count] = null;
                System.out.println("\nItem Removed");
                return;
            }
        }
        System.out.println("Item not Found");
    }

}
```

```

    double total()
    {
        double sum = 0;
        for(int i= 0;i<count;i++)sum += items[i].price;
        return sum;
    }

    void display() {
        if(count == 0)
        {
            System.out.println("Cart empty");
            return;
        }
        for(int i = 0;i<count;i++)
        {
            System.out.println(items[i].name+ " "+items[i].price);
            System.out.println("Total: " +total());
        }
    }
}

```

Main:

```

package Online_ShoppingCartSystem;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

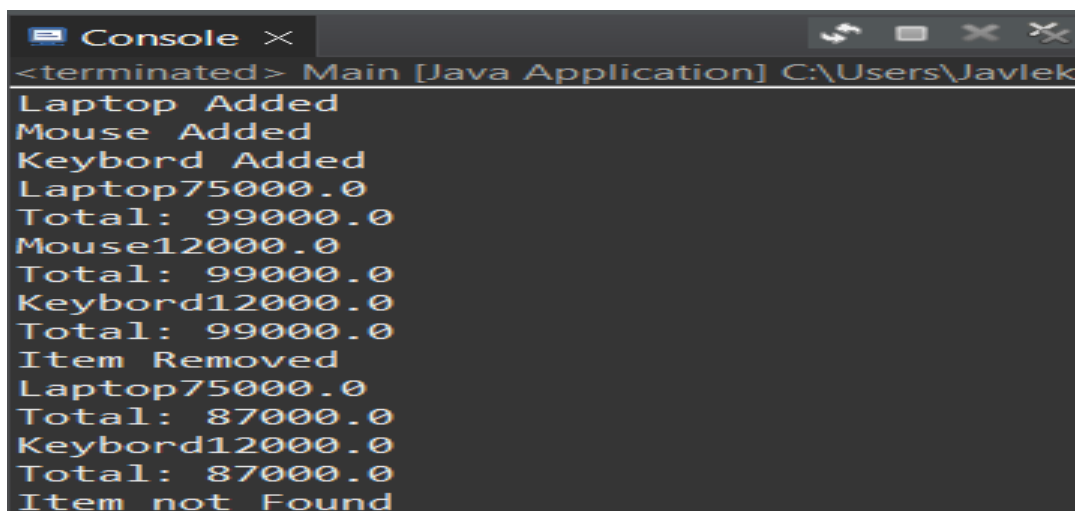
        Cart c = new Cart();
        c.addItem(new Product(1,"Laptop",75000));
        c.addItem(new Product(2,"Mouse",12000));
        c.addItem(new Product(3,"Keyboard",12000));
        c.display();
        c.removeItem(2);
        c.display();
        c.removeItem(4);

    }

}

```

Output:



```

<terminated> Main [Java Application] C:\Users\Javlek
Laptop Added
Mouse Added
Keyboard Added
Laptop75000.0
Total: 99000.0
Mouse12000.0
Total: 99000.0
Keyboard12000.0
Total: 99000.0
Item Removed
Laptop75000.0
Total: 87000.0
Keyboard12000.0
Total: 87000.0
Item not Found

```


Q6. Car Rental System (Interface Implementation):

Define an interface RentalService with a method calculateRentalPrice(int days). Implement two classes CarRental and BikeRental with different rental rates per day. Use polymorphism to calculate rental prices dynamically based on user input.

Rental Service:

```
package Car_RentalSystem;

public interface RentalService {

    int calculateRentalPrice(int days);

}
```

Car Rental:

```
package Car_RentalSystem;

public class CarRental implements RentalService {

    @Override
    public int calculateRentalPrice(int days) {

        return 3000 * days;

    }

}
```

Bike Rental:

```
package Car_RentalSystem;

public class BikeRental implements RentalService {

    @Override
    public int calculateRentalPrice(int days) {

        return 1000 * days;

    }

}
```

Rental System:

```
package Car_RentalSystem;

import java.util.Scanner;

public class RentalSystem {

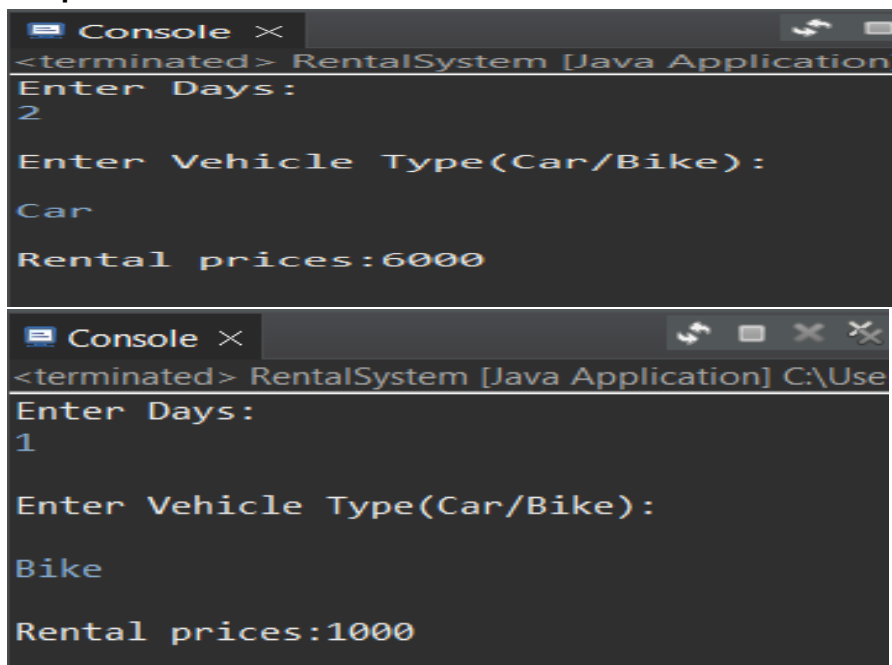
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Days: ");
        int days = sc.nextInt();

        System.out.println("\nEnter Vehicle Type(Car/Bike): ");
        String vehicleType = sc.next().toLowerCase();

        if(vehicleType.equals("car"))
        {
            CarRental car = new CarRental();
            int fare = car.calculateRentalPrice(days);
            System.out.println("\nRental prices:" +fare);
        }
        else if(vehicleType.equals("bike"))
        {
            BikeRental bike = new BikeRental();
            int fare = bike.calculateRentalPrice(days);
            System.out.println("\nRental prices:" +fare);
        }
        else
        {
            System.out.println("\nUser Entered vehicle is not valid!");
        }
    }
}
```

Output:



```
Console ×
<terminated> RentalSystem [Java Application]
Enter Days:
2
Enter Vehicle Type(Car/Bike):
Car
Rental prices:6000

Console ×
<terminated> RentalSystem [Java Application] C:\Use
Enter Days:
1
Enter Vehicle Type(Car/Bike):
Bike
Rental prices:1000
```

Q7. Age Verification for Driving License

Ask the user to enter their age.

If the age is less than 18, throw an InvalidAgeException.

Display "You must be 18 or older to apply for a driving license"

Invalid Age Exception:

```
package Age_VerificationforDrivingLicense;

public class InvalidAgeException extends Exception {

    public InvalidAgeException(String message) {
        super(message);
    }

}
```

Age Check:

```
package Age_VerificationforDrivingLicense;

import java.util.Scanner;

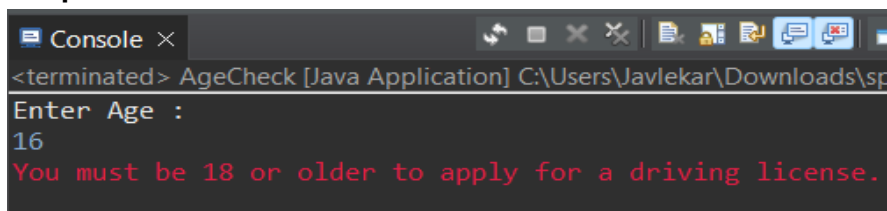
public class AgeCheck {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

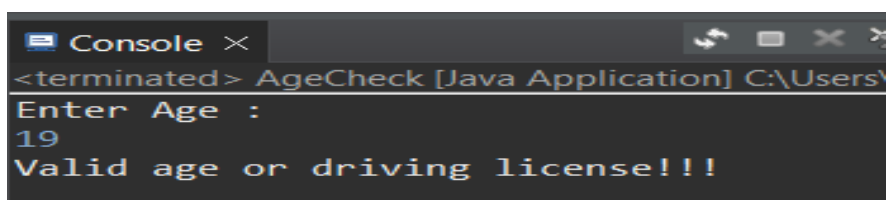
        System.out.println("Enter Age : ");
        int age = sc.nextInt();

        try
        {
            if (age < 18)
            {
                throw new InvalidAgeException("You must be 18 or older to apply for a driving license.");
            }
            System.out.println("Valid age or driving license!!!");
        }
        catch (InvalidAgeException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

Output:



```
Console ×
<terminated> AgeCheck [Java Application] C:\Users\Javlekar\Downloads\sp
Enter Age :
16
You must be 18 or older to apply for a driving license.
```



```
Console ×
<terminated> AgeCheck [Java Application] C:\Users\
Enter Age :
19
Valid age or driving license!!!
```

Q8. Division Calculator (Handling Divide by Zero)

Write a program that asks for two numbers and performs division.

If the denominator is zero, handle `ArithmeticException` and display "Division by zero is not allowed".

```
package Division_Calculator;

import java.util.Scanner;

public class HandleZero {

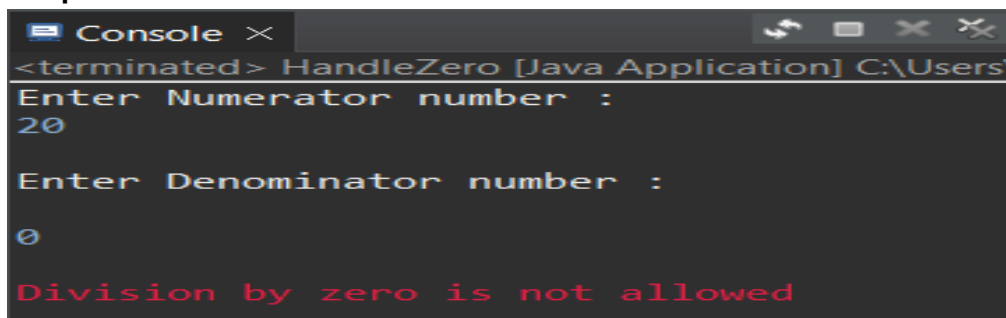
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Numerator number : ");
        int numerator = sc.nextInt();

        System.out.println("\nEnter Denominator number : ");
        int denominator = sc.nextInt();

        try {
            if(denominator == 0) {
                throw new ArithmeticException("\nDivision by zero is not allowed");
            }
            int result = numerator / denominator;
            System.out.println("\nResult : " + result);
        } catch (ArithmeticException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

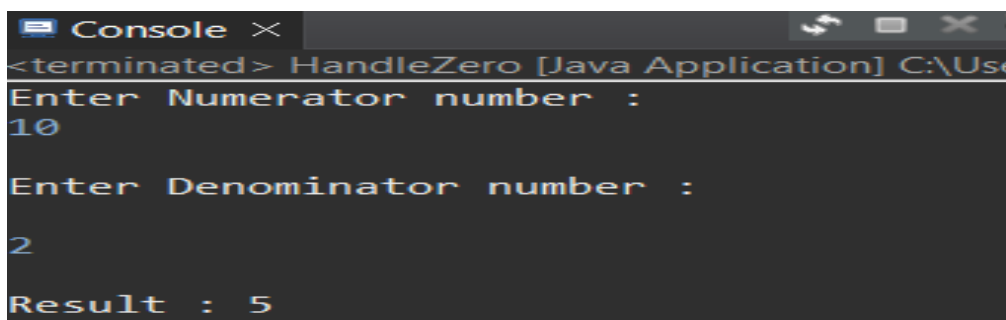
Output:



```
Console x [Java Application] C:\Users\
<terminated> HandleZero [Java Application] C:\Users\
Enter Numerator number :
20

Enter Denominator number :
0

Division by zero is not allowed
```



```
Console x [Java Application] C:\Use
<terminated> HandleZero [Java Application] C:\Use
Enter Numerator number :
10

Enter Denominator number :
2

Result : 5
```

Q9. Online Shopping - Product Availability

Create an array of 5 products (["Laptop", "Phone", "Tablet", "Watch", "Headphones"]).

Ask the user for a product index (0-4).

If the user enters an invalid index, handle `ArrayIndexOutOfBoundsException` and display "Product not found".

Array Index Out Of Bounds Exception:

```
package OnlineShopping_ProductAvailability;

public class ArrayIndexOutOfBoundsException extends Exception {

    public ArrayIndexOutOfBoundsException(String message) {

        super(message);
    }
}
```

Product Availability:

```
package OnlineShopping_ProductAvailability;

import java.util.Scanner;

public class ProductAvailability {

    public static void main(String[] args) {

        String[] product = {"Laptop", "Phone", "Tablet", "Watch", "Headphones"};

        Scanner sc = new Scanner(System.in);

        try
        {
            System.out.println("Enter product index (0-4): ");
            int index = sc.nextInt();
            System.out.println("Product at index " +index+ " : " +product[index]);
        }
        catch(Exception e)
        {
            System.err.println("Product Not Found");
        }
    }
}
```

Output:

```
Console × [ProductAvailability Java Application]
<terminated> ProductAvailability [Java Application]
Enter product index (0-4):
6
Product Not Found
```

```
Console × [ProductAvailability Java Appl]
<terminated> ProductAvailability [Java Appl]
Enter product index (0-4):
3
Product at index 3 : Watch
```

Q10. Invalid Bank Account Number

Ask the user to enter a 6-digit bank account number.

If the entered number has less/more than 6 digits, throw

InvalidAccountNumberException.

Display "Please enter a valid 6-digit account number".

Invalid Account Number Exception:

```
package Invalid_BankAccountNumber;

public class InvalidAccountNumberException extends Exception {

    public InvalidAccountNumberException(String message) {

        super(message);

    }

}
```

Invalid Account Number:

```
package Invalid_BankAccountNumber;

import java.util.Scanner;

public class InvalidAccountNumber {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter 6 Digit Account Number: ");
        String AccountNumber = sc.next();

        try {

            if(AccountNumber.length() != 6)

            {

                throw new InvalidAccountNumberException("Please enter a valid 6-digit account number.");

            }

            System.out.println("Valid Account Number");

        }

        catch(InvalidAccountNumberException e)

        {

            System.err.println(e.getMessage());

        }

    }

}
```

Output:

```
Console ×
<terminated> InvalidAccountNumber [Java Application] C:\User
Enter 6 Digit Account Number:
546
Please enter a valid 6-digit account number.
```

```
Console ×
<terminated> InvalidAccountNumber [Java Application]
Enter 6 Digit Account Number:
456875
Valid Account Number
```