Introduction to Object Oriented Programming

Workshop 5

Abstraction

1. **Create an abstract class Shape**

2. The Shape class has two abstract methods: calculateArea()

and calculatePerimeter. Both the methods have a return type

of void

3. Create

a class Quadrilateral which extends the abstract class

Shape.

4. Implement all the abstract method of the parent class

Code:

```java
abstract class shape{
    abstract void calculateArea();
    abstract void calculatePerimeter();
}
class Quadrilateral extends shape{
    int A;
    int B;
    int C;
    int D;
    Quadrilateral(int A,int B,int C,int D){
        this.A=A;
        this.B=B;
        this.C=C;
        this.D=D;
    }
    @Override
    void calculateArea(){
       int area=A+B+C+D;
       System.out.println(area);
    }
    @Override
    void calculatePerimeter(){
        int perimeter=(A+B+C+D)/2;
        System.out.println(perimeter);
    }
}
```

```java
public class workshop5 {
    Run | Debug
    public static void main(String[] args) {
        Quadrilateral quadrilateral=new Quadrilateral(A:50, B:5
        quadrilateral.calculateArea();
        quadrilateral.calculatePerimeter();
    }
}
```

Output:

```
review' '-XX:+ShowCod
_ws\workshop5_f1ff0b5
200
100
PS C:\Users\user\OneD
```

5. Create an abstract class named Vehicle which consist of two methods: wheel and door. Both the methods have void return type and no parameters. The method wheel has no implementation.

6. Create a class name Bus and extend the Vehicle class.

Interface

Code:

```java
abstract class vehicle{
    abstract void wheel();
    abstract void door();
}
class Bus extends vehicle{
    void wheel(){
      System.out.println(x:"it is wheel of the bus");
    }
    void door(){
       System.out.println(x:"it is door of the bus");
    }
}

public class workshop5 {
    Run | Debug
    public static <B> void main(String[] args) {
        Bus bus=new Bus();
        bus.wheel();
        bus.door();
    }
}
```

Output:

```
review' '-XX:+ShowCodeDetailsInE
_ws\workshop5_f1ff0b56\bin' 'wor
it is wheel of the bus
it is door of the bus
PS C:\Users\user\OneDrive\Deskto
```

7. Create an interface Animal. The Animal interface has two methods eat() and walk()

8. Create another interface Printable. The Printable interface has a method called display();

9. Create a class Cow that implements the Animal and Printable interfaces

10. Create an interface LivingBeing

11. Create an method void specialFeature()

12. Create 2 classes Fish and Bird that implements LivingBeing

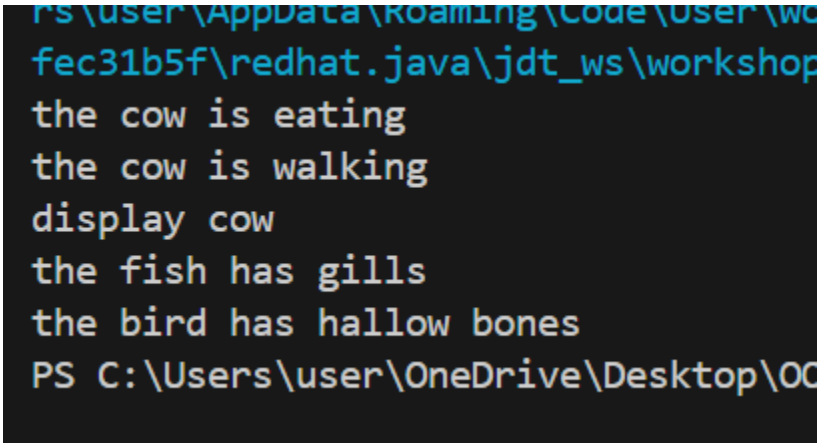
13. The specialFeature should display special features of the respective class animal.

Code:

```java
interface Animal{
    void eat();
    void walk();
}
interface printable{
    void display();
}
interface livingBeing{
    void specialFeature();
}
class Cow implements Animal,printable{
    @Override
    public void eat(){
        System.out.println(x:"the cow is eating");
    }
    @Override
    public void walk(){
        System.out.println(x:"the cow is walking");
    }
    @Override
    public void display(){
        System.out.println(x:"display cow");
    }
}
```

```java
}
class fish implements livingBeing{
    @Override
    public void specialFeature(){
        System.out.println(x:"the fish has gills");
    }
}
class bird implements livingBeing{
    @Override
    public void specialFeature(){
        System.out.println(x:"the bird has hallow bones");
    }
}
public class workshop5 {
    Run | Debug
    public static void main(String[] args) {
        Cow cow =new Cow();
        cow.eat();
        cow.walk();
        cow.display();
        fish Fish=new fish();
        Fish.specialFeature();
        bird Bird=new bird();
        Bird.specialFeature();
    }
}
```

Output:



Exception

14. In the following program, which exception will be generated

```
public class Demo{
public static void main(String[] args) {
System.out.println(10/0);
}
}
```

Handle the exception above by using try-catch.

Code:

```java
public class workshop5{
    Run | Debug
    public static void main(String[] args) {
    try{
     System.out.println(10/0);
    }catch (Exception e){
        System.out.println(e);
    }
}

}
```

Output:

```
review' '-XX:+ShowCodeDetailsInExceptionMes
_ws\workshop5_f1ff0b56\bin' 'workshop5'
java.lang.ArithmeticException: / by zero
```

15. In the following program, which exception will be

generated

public class Demo{

public static void main(String[] args) {

int[] age = {10,20,25,24,28,27,30,31,32};

System.out.println(age[9]);

}

}

Handle the exception by using throws keyword

Code:

```java
public class workshop5{
    Run | Debug
    public static void main(String[] args) {
    int[] age = {10,20,25,24,28,27,30,31,32};
    int index=9;
    try {
        if(index>= age.length){
            throw new ArrayIndexOutOfBoundsException(s:"index
        }
        System.out.println(age[index]);
    } catch (ArrayIndexOutOfBoundsException e) {
        throw new ArrayIndexOutOfBoundsException(s:"\n custom
    }
    System.out.println(age[9]);
    }
    }
```

Output:

```
_ws\workshop5_f1ff0b56\bin' 'workshop5'
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsExcept
 custom message : index out of bounds
        at workshop5.main(workshop5.java:126)
```