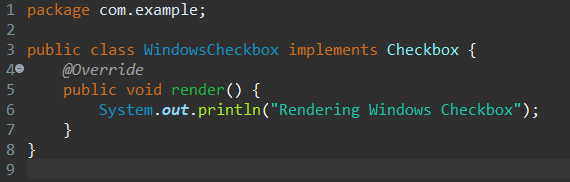
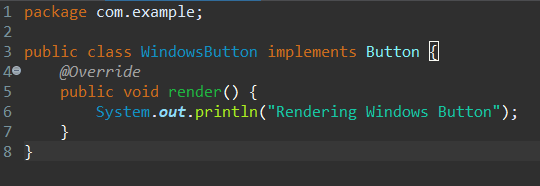
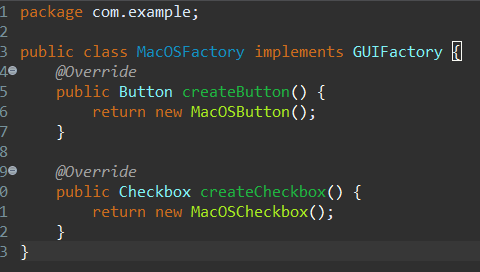
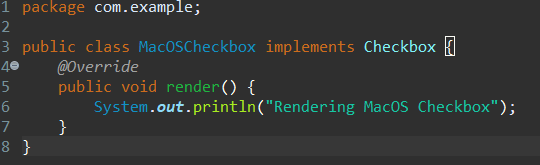
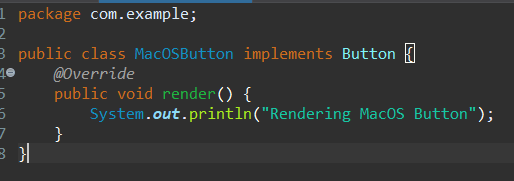
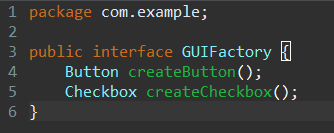
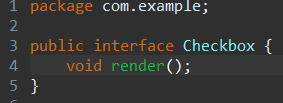
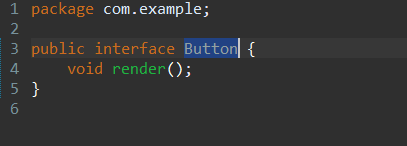
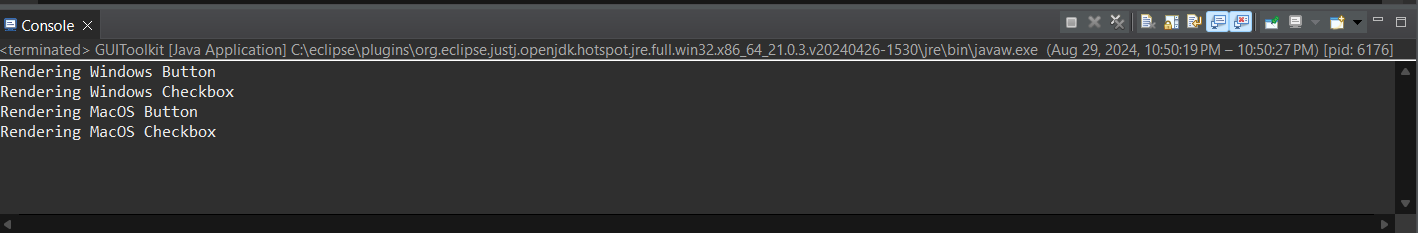
**Module\_1\_Assignment:**

1. Design a GUI toolkit using Abstract Factory design pattern that should support multiple operating systems, such as Windows and MacOS. Each OS has its own way of rendering UI components like buttons and checkboxes. Illustrate its components.

CODE:

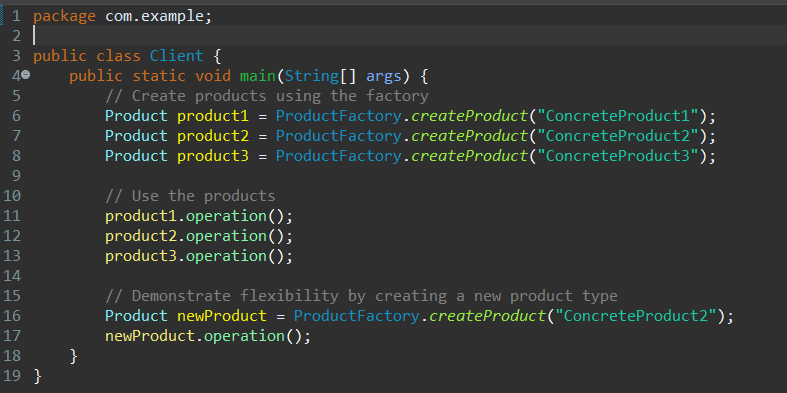
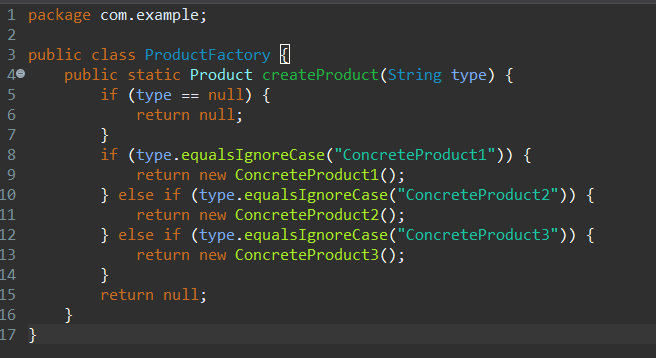
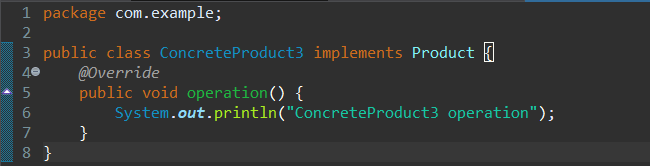
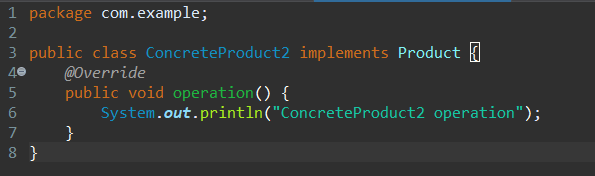
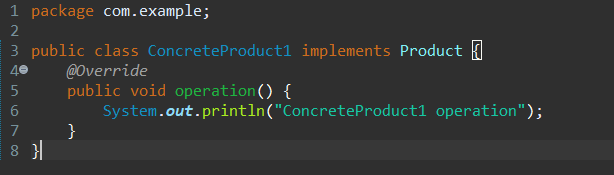
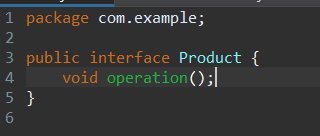


OUTPUT:

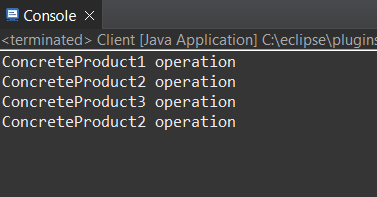


1. Write a Java program implementing the Factory Design Pattern with one interface (Product) and three classes (ConcreteProduct1, ConcreteProduct2, and ConcreteProduct3). Include a ProductFactory for object creation based on types. Demonstrate the pattern's flexibility in the main method of a Client class and highlight its benefits.

CODE:



OUTPUT:



1. Differentiate interface with abstract class in java.

ANSWER:

**Interface:**

* An interface is a abstract class that contains only constants and method signatures (no implementation).
* All methods in an interface are implicitly abstract and public.
* A class that implements an interface must provide an implementation for all the methods in the interface.
* Multiple inheritance is allowed in interfaces, meaning a class can implement multiple interfaces.
* Interfaces are used to define a contract that must be implemented by any class that implements it.

**Abstract Class:**

* An abstract class is a class that cannot be instantiated and is designed to be inherited by other classes.
* An abstract class can have both abstract and non-abstract methods (i.e., methods with implementation).
* A class that extends an abstract class must provide an implementation for all the abstract methods in the abstract class.
* Multiple inheritance is not allowed in abstract classes, meaning a class can only extend one abstract class.
* Abstract classes are used to provide a partial implementation of a class that can be shared by multiple subclasses.

// Interface

public interface Flyable {

void fly();

}

// Abstract Class

public abstract class Bird {

public abstract void makeSound();

public void eat() {

System.out.println("Eating...");

}

}

// Class implementing the interface

public class Parrot implements Flyable {

public void fly() {

System.out.println("Flying...");

}

}

// Class extending the abstract class

public class Sparrow extends Bird {

public void makeSound() {

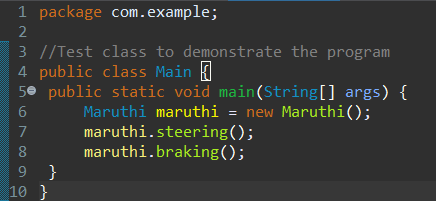
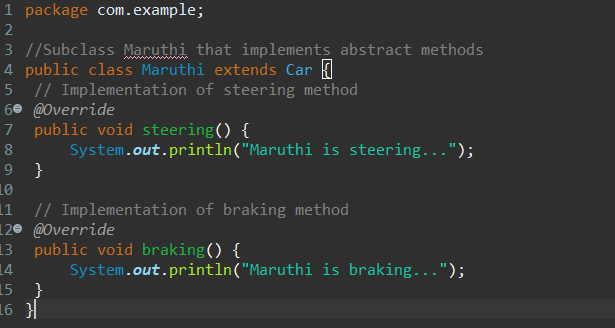
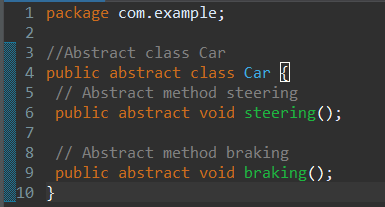
System.out.println("Chirping...");

}

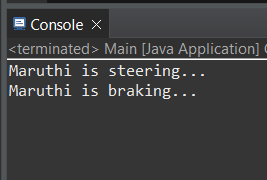
}

1. Build a Java program to create abstract class Car, with abstract methods steering () and braking (). Create a subclass Maruthi to implements abstract methods in it.

CODE:



CODE:



1. Analyze the Data class code smell, characterized by the declaration and use of only instance variables without any accompanying methods interacting with other classes. Provide a concise Java example illustrating this code smell.

ANSWER:

Data Class is a class that contains only instance variables and no methods, essentially making it a dumb data holder. This code smell is often characterized by the lack of behavior or functionality in a class, making it a mere data container.

public class Person {

public String firstName;

public String lastName;

public int age;

public String address;

}

This code smell can lead to issues such as:

* Tight coupling: Other classes may need to access and manipulate the data directly, leading to tight coupling between classes.
* Lack of encapsulation: The data is exposed publicly, making it vulnerable to external modifications.
* Inflexibility: The class is not extensible, and any changes to the data or behavior would require modifications to the class itself.