1. Command line arguments

```
public class Exp2 {
    public static void main(String[] args) {
        if (args.length > 0) {
            System.out.println("Command Line Arguments:");
            for (String arg : args) {
                System.out.println(arg);
            }
        } else {
            System.out.println("No command line arguments provided.");
        }
    }
}
```

2. Method overloading

```
public class MethodOverloading {
    public static void main(String[] args) {
        Calculator calculator = new Calculator();

        System.out.println("Method Overloading :");
        System.out.println("Adding 2 variables: " + calculator.add(5,10));
        System.out.println("Adding after changing data type of variables:"
+ calculator.add(5.5, 10.5));
        System.out.println("Adding 3 variables: " + calculator.add(5, 10,
15));
    }
}
class Calculator {
    int add(int a, int b) { return a + b; }
    double add(double a, double b) { return a + b; }
    int add(int a, int b, int c) { return a + b + c; }
}
```

3. Method overriding

```
import java.util.Scanner;

class Vehicle {
    void start() {
        System.out.println("Vehicle is starting");
    }
}

class Car extends Vehicle {
    @Override
    void start() {
        System.out.println("Car is starting");
    }
}

public class MethodOverriding {
    public static void main(String[] args) {
        Vehicle vehicle = new Car();
        vehicle.start();
    }
}
```

4. Abstract class

```
import java.util.Scanner;

abstract class Shape {
    abstract void draw();
}

class Circle extends Shape {
    void draw() {
        System.out.println("Drawing a circle");
    }
}

public class Abs {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Shape shape = new Circle();
        shape.draw();
        scanner.close();
    }
}
```

5. Wrapper class

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

        System.out.print("Enter an integer: ");
        int num = scanner.nextInt();

        Integer wrappedNum = num;
        System.out.println("Wrapped Integer: " + wrappedNum);
        int unwrappedNum = wrappedNum;
        System.out.println("Unwrapped Integer: " + unwrappedNum);
        scanner.close();
    }
}
```

6. Encapsulation

```
import java.util.Scanner;

class Encap {
    private String data;
    void setData(String value) {
        data = value;
    }
    String getData() {
        return data;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Encap obj = new Encap();
        System.out.print("Enter data: ");
        obj.setData(scanner.nextLine());
        System.out.println("Data: " + obj.getData());
        scanner.close();
    }
}
```

7. Single Inheritance

```
import java.util.Scanner;
class Parent {
    void parentMethod() {
        System.out.println("Parent method");
    }
}
class Child extends Parent {
    void childMethod() {
        System.out.println("Child method");
    }
}
public class Single {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Child child = new Child();
        child.parentMethod();
        child.childMethod();
        scanner.close();
    }
```

```java
}
```

8. Multilevel inheritance

```java
class A {
    void methodA() {
        System.out.println("Method in A");
    }
}

class B extends A {
    void methodB() {
        System.out.println("Method in B");
    }
}

class C extends B {
    void methodC() {
        System.out.println("Method in C");
    }
}

public class Multi {
    public static void main(String[] args) {
        C c = new C();
        c.methodA();
        c.methodB();
        c.methodC();
    }
}
```

9. Interface

```java
interface Shape {
    double calculateArea();
}

class Circle implements Shape {
    private double r;
    public Circle(double r) { this.r = r; }
    public double calculateArea() { return Math.PI * r * r; }
}

public class InterfaceExample {
    public static void main(String[] args) {
        Circle circle = new Circle(5.0);
        System.out.println("Area of the circle: " +
circle.calculateArea());
    }
}
```

10. Exception handling

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

public class Exp11 {
    public static void main(String[] args) {
        try {
            int age = 15;

            if (age < 18) {
                throw new CustomException("Age must be 18 or older to
access this service.");
            }

            System.out.println("Access granted. Welcome!");
        } catch (CustomException e) {
            System.out.println("Custom Exception: " + e.getMessage());
        }
    }
}
```

11. Multithreading

```
public class MultithreadingExample {
    public static void main(String[] args) {
        Thread thread1 = new Thread(new MyRunnable("Thread 1"));
        Thread thread2 = new Thread(new MyRunnable("Thread 2"));
        thread1.start();
        thread2.start();
    }
}

class MyRunnable implements Runnable {
    private String threadName;

    public MyRunnable(String name) {
        this.threadName = name;
    }

    @Override
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(threadName + ": Count " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```