Name: Nandeesh N

1.

```
package TypeCasting;

public class TypeCasting {
    public static void main(String[] args) {
        // Implicit Type Casting
        int x = 10;
        double y = x; // Implicit casting of int to double
        System.out.println("Implicit Casting: int to double: " + y);

        // Explicit Type Casting
        double a = 10.5;
        int b = (int) a; // Explicit casting of double to int
        System.out.println("Explicit Casting: double to int: " + b);
    }
}
```

2.Access Modifiers

```
package AccessModifiers;

public class AccessModifiers {
 public static void main(String[] args) {
 // Create an object of AnotherClass
 AnotherClass obj = new AnotherClass();
 obj.display();
 }
}

class AnotherClass {
 private void display() {
 System.out.println("Private method can be accessed only within the class.");
 }
}
```

3.While Loop

```
package Loops;

public class WhileLoop {
    public static void main(String[] args) {
        int i = 1;
        while (i <= 5) {
```

```
            System.out.println("While loop iteration: " + i);
            i++;
        }
    }
}
```

4.Do While Loop

```
package Loops;

public class DoWhileLoop {
    public static void main(String[] args) {
        int i = 1;
        do {
            System.out.println("Do-While loop iteration: " + i);
            i++;
        } while (i <= 5);
    }
}
```

5.For Loop

```
package Loops;

public class ForLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            System.out.println("For loop iteration: " + i);
        }
    }
}
```

6.Classes, Objects, Constructors, and Inheritance

```
package Inheritance;

public class Animal {
    public Animal() {
        System.out.println("Constructor of Animal class.");
    }

    public void sound() {
        System.out.println("Animals make different sounds.");
    }
```

```
}

class Dog extends Animal {
    public Dog() {
        System.out.println("Constructor of Dog class.");
    }

    public void sound() {
        System.out.println("Dog barks.");
    }
}

class InheritanceDemo {
    public static void main(String[] args) {
        Dog myDog = new Dog();
        myDog.sound();
    }
}
```

6.Collections Implementation

```
package Collections;

import .util.ArrayList;

public class CollectionsExample {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();

        // Adding elements to the ArrayList
        list.add("Apple");
        list.add("Banana");
        list.add("Orange");

        // Displaying elements
        System.out.println("ArrayList Elements: " + list);
    }
}
```

11.Try-Catch Block

```
package ExceptionHandling;

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

```
            int num = 10 / 0; // This line will throw an ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println("Caught an ArithmeticException: " + e.getMessage());
        }
    }
}
```

Throw and Throws Keyword

```
package ExceptionHandling;

import .io.IOException;

public class ThrowAndThrows {
    public static void main(String[] args) {
        try {
            validateAge(15);
        } catch (IOException e) {
            System.out.println("Caught an IOException: " + e.getMessage());
        }
    }

    public static void validateAge(int age) throws IOException {
        if (age < 18) {
            throw new IOException("Age must be greater than or equal to 18.");
        } else {
            System.out.println("Valid age.");
        }
    }
}
```

12.Try Block with Parameters

```
package ExceptionHandling;

public class TryBlockWithParameters {
    public static void main(String[] args) {
        try {
            int result = divideNumbers(10, 0);
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Caught an ArithmeticException: " + e.getMessage());
        }
    }

    public static int divideNumbers(int a, int b) {
        return a / b;
```

```
    }
}
```

11,Multiple Catch Blocks

```
package ExceptionHandling;

public class MultipleCatchBlocks {
    public static void main(String[] args) {
        try {
            int[] array = new int[5];
            array[5] = 10 / 0; // This line will throw an ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println("Caught an ArithmeticException: " + e.getMessage());
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Caught an ArrayIndexOutOfBoundsException: " +
e.getMessage());
        } catch (Exception e) {
            System.out.println("Caught an Exception: " + e.getMessage());
        }
    }
}
```

14.Finally Block

```
.package ExceptionHandling;

public class FinallyBlock {
    public static void main(String[] args) {
        try {
            int result = divideNumbers(10, 0);
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Caught an ArithmeticException: " + e.getMessage());
        } finally {
            System.out.println("Finally block executed.");
        }
    }

    public static int divideNumbers(int a, int b) {
        return a / b;
    }
}ects, Constructors, and Inheritance
```

```
package Inheritance;

public class Animal {
 public Animal() {
 System.out.println("Constructor of Animal class.");
 }

 public void sound() {
 System.out.println("Animals make different sounds.");
 }
}

class Dog extends Animal {
 public Dog() {
 System.out.println("Constructor of Dog class.");
 }

 public void sound() {
 System.out.println("Dog barks.");
 }
}

class InheritanceDemo {
 public static void main(String[] args) {
 Dog myDog = new Dog();
 myDog.sound();
 }
}
```

Collections Implementation
For demonstration of collections, various collections such as ArrayList, HashMap, etc., can be used based on specific requirements. Here's a basic example using ArrayList:

```
package Collections;

import java.util.ArrayList;

public class CollectionsExample {
 public static void main(String[] args) {
 ArrayList<String> list = new ArrayList<>();

 // Adding elements to the ArrayList
 list.add("Apple");
 list.add("Banana");
 list.add("Orange");
```

```
 // Displaying elements
 System.out.println("ArrayList Elements: " + list);
 }
}
```

Try-Catch Block

```
package ExceptionHandling;

public class TryCatchBlock {
 public static void main(String[] args) {
 try {
 int num = 10 / 0; // This line will throw an ArithmeticException
 } catch (ArithmeticException e) {
 System.out.println("Caught an ArithmeticException: " + e.getMessage());
 }
 }
}
```

Throw and Throws Keyword

```
package ExceptionHandling;

import .io.IOException;

public class ThrowAndThrows {
 public static void main(String[] args) {
 try {
 validateAge(15);
 } catch (IOException e) {
 System.out.println("Caught an IOException: " + e.getMessage());
 }
 }

 public static void validateAge(int age) throws IOException {
 if (age < 18) {
 throw new IOException("Age must be greater than or equal to 18.");
 } else {
 System.out.println("Valid age.");
 }
 }
}
```

Try Block with Parameters

In , try blocks cannot take parameters directly. However, a method within a try block can take parameters.

Here's a simple illustration:

```
package ExceptionHandling;

public class TryBlockWithParameters {
 public static void main(String[] args) {
 try {
 int result = divideNumbers(10, 0);
 System.out.println("Result: " + result);
 } catch (ArithmeticException e) {
 System.out.println("Caught an ArithmeticException: " + e.getMessage());
 }
 }

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

Multiple Catch Blocks

```
package ExceptionHandling;

public class MultipleCatchBlocks {
 public static void main(String[] args) {
 try {
 int[] array = new int[5];
 array[5] = 10 / 0; // This line will throw an ArithmeticException
 } catch (ArithmeticException e) {
 System.out.println("Caught an ArithmeticException: " + e.getMessage());
 } catch (ArrayIndexOutOfBoundsException e) {
 System.out.println("Caught an ArrayIndexOutOfBoundsException: " + e.getMessage());
 } catch (Exception e) {
 System.out.println("Caught an Exception: " + e.getMessage());
 }
 }
}
```

Finally Block

```
package ExceptionHandling;

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

```
 int result = divideNumbers(10, 0);
 System.out.println("Result: " + result);
 } catch (ArithmeticException e) {
 System.out.println("Caught an ArithmeticException: " + e.getMessage());
 } finally {
 System.out.println("Finally block executed.");
 }
 }

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