1. Write a Program for Widening, Narrowing, and NumberFormatException

```
import java.util.Scanner;

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

        intValue = sc.nextInt();
        double widened = intValue;
        System.out.println(intValue);
        System.out.println(widened);

        double doubleValue = sc.nextDouble();
        int narrowed = (int) doubleValue;
        short shortValue = (short) narrowed;
        System.out.println(narrowed);
        System.out.println(shortValue);

        try {
            int number = 123;
            String str = String.valueOf(number);
            System.out.println(str);
            int parsed = Integer.parseInt(str);
            System.out.println(parsed);
        } catch (NumberFormatException e) {
            System.out.println("Invalid number format");
        }
    }
}
```

Sample Output

```
5
5.0
```

```
7.9
7
7
123
123
```

2. Write a Program for Compound Assignment Behaviour

```
public class CompoundAssignment {
    public static void main(String[] args) {
        int x = 5;
        // x = x + 4.5; // compile error
        x += 4.5;
        System.out.println(x);
    }
}
```

Sample Output

```
9
```

3. Write a Program for Object Casting with Inheritance

```
class Animal {
    void makeSound() {
        System.out.println("Animal sound");
    }
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Woof!");
    }
    void fetch() {
        System.out.println("Dog fetches ball");
    }
}

public class CastingDemo {
    public static void main(String[] args) {
        Dog d = new Dog();
        Animal a = d;
        a.makeSound();
    }
}
```

Sample Output

```
Woof!
```

---

4. Write a Program for Temperature Converter class

```
import java.util.Scanner;

public class TempConverter {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double celsius = sc.nextDouble();
        double fahrenheit = celsius * 9 / 5 + 32;
        int truncated = (int) fahrenheit;
        System.out.println(fahrenheit);
        System.out.println(truncated);
    }
}
```

Sample Output

```
37
98
```

---

5. Write a Program for Enum – Days of the Week

```
import java.util.Scanner;
```

```
enum DaysOfWeek {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
}

public class DaysEnumDemo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String dayName = sc.next().toUpperCase();
        try {
            DaysOfWeek day = DaysOfWeek.valueOf(dayName);
            System.out.println(day.ordinal());
            if (day == DaysOfWeek.SATURDAY || day == DaysOfWeek.SUNDAY) {
                System.out.println("Weekend");
            } else {
                System.out.println("Weekday");
            }
        } catch (IllegalArgumentException e) {
            System.out.println("Invalid day");
        }
    }
}
```

Sample Output

```
SUNDAY
6
Weekend
```

---

6. Write a Program for Enum – Compass Directions

```
import java.util.Scanner;

enum Direction {
    NORTH, SOUTH, EAST, WEST
}
```

```
public class DirectionDemo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String input = sc.next().toUpperCase();
        try {
            Direction dir = Direction.valueOf(input);
            switch (dir) {
                case NORTH -> System.out.println("Move north");
                case SOUTH -> System.out.println("Move south");
                case EAST -> System.out.println("Move east");
                case WEST -> System.out.println("Move west");
            }
        } catch (IllegalArgumentException e) {
            System.out.println("Invalid direction");
        }
    }
}
```

Sample Output

```
EAST
Move east
```

---

7. Write a Program for Enum – Shape Area Calculator

```
enum Shape {
    CIRCLE {
        double area(double... params) { return Math.PI * params[0] * params[0]; }
    },
    SQUARE {
        double area(double... params) { return params[0] * params[0]; }
    },
    RECTANGLE {
        double area(double... params) { return params[0] * params[1]; }
    },
    TRIANGLE {
        double area(double... params) { return 0.5 * params[0] * params[1]; }
    };
    abstract double area(double... params);
}

public class ShapeDemo {
    public static void main(String[] args) {
        System.out.println("Circle: " + Shape.CIRCLE.area(5));
        System.out.println("Square: " + Shape.SQUARE.area(4));
        System.out.println("Rectangle: " + Shape.RECTANGLE.area(4, 6));
        System.out.println("Triangle: " + Shape.TRIANGLE.area(4, 6));
    }
}
```

Sample Output

```
Circle: 78.53981633974483
Square: 16.0
Rectangle: 24.0
Triangle: 12.0
```

---

8. Write a Program for Enum – Card Suit & Rank with Deck

```
import java.util.*;
enum Suit { CLUBS, DIAMONDS, HEARTS, SPADES }
enum Rank { ACE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK,
QUEEN, KING }

class Card {
    Suit suit;
    Rank rank;
    Card(Suit s, Rank r) { suit = s; rank = r; }
    public String toString() { return rank + " of " + suit; }
}

class Deck {
    List<Card> cards = new ArrayList<>();
    Deck() {
        for (Suit s : Suit.values()) {
            for (Rank r : Rank.values()) {
                cards.add(new Card(s, r));
            }
        }
    }
    void shuffle() { Collections.shuffle(cards); }
    void printDeck() { for (Card c : cards) System.out.println(c); }
}

public class CardDeckDemo {
    public static void main(String[] args) {
        Deck deck = new Deck();
        deck.shuffle();
        deck.printDeck();
    }
}
```

Sample Output (partial)

```
NINE of DIAMONDS
QUEEN of SPADES
THREE of HEARTS
```

---

9. Write a Program for Enum – Priority Levels

```
enum PriorityLevel {
    LOW(1), MEDIUM(2), HIGH(3), CRITICAL(4);
    int code;
    PriorityLevel(int code) { this.code = code; }
    boolean isUrgent() { return code >= 3; }
}

public class PriorityDemo {
    public static void main(String[] args) {
        for (PriorityLevel p : PriorityLevel.values()) {
            System.out.println(p + " Code: " + p.code + " Urgent: " + p.isUrgent());
        }
    }
}
```

Sample Output

```
LOW Code: 1 Urgent: false
MEDIUM Code: 2 Urgent: false
HIGH Code: 3 Urgent: true
CRITICAL Code: 4 Urgent: true
```

---

10. Write a Program for Enum – Traffic Light State Machine

```
interface State { State next(); }
enum TrafficLight implements State {
    RED {
        public State next() { return GREEN; }
    },
    GREEN {
        public State next() { return YELLOW; }
    },
    YELLOW {
        public State next() { return RED; }
    }
}
public class TrafficLightDemo {
    public static void main(String[] args) {
        State state = TrafficLight.RED;
        for (int i = 0; i < 6; i++) {
            System.out.println(state);
            state = state.next();
        }
    }
}
```

Sample Output

```
RED
GREEN
YELLOW
RED
GREEN
YELLOW
```

---

11. Write a Program for Enum – Difficulty Level & Game Setup

```
enum Difficulty { EASY, MEDIUM, HARD }
class Game {
    Game(Difficulty diff) {
        switch (diff) {
            case EASY -> System.out.println("3000 bullets");
            case MEDIUM -> System.out.println("2000 bullets");
            case HARD -> System.out.println("1000 bullets");
        }
    }
}

public class GameDemo {
    public static void main(String[] args) {
        new Game(Difficulty.EASY);
        new Game(Difficulty.HARD);
    }
}
```

Sample Output

```
3000 bullets
1000 bullets
```

---

12. Write a Program for Enum – Calculator Operations

```
enum Operation {
    PLUS { double eval(double a, double b) { return a + b; } },
    MINUS { double eval(double a, double b) { return a - b; } },
    TIMES { double eval(double a, double b) { return a * b; } },
    DIVIDE { double eval(double a, double b) { return a / b; } };
    abstract double eval(double a, double b);
}

public class OperationDemo {
    public static void main(String[] args) {
        System.out.println(Operation.PLUS.eval(5, 3));
        System.out.println(Operation.DIVIDE.eval(10, 2));
    }
}
```

Sample Output

```
8.0
5.0
```

---

13. Write a Program for Enum – Knowledge Level

```
enum KnowledgeLevel {
    BEGINNER, ADVANCED, PROFESSIONAL, MASTER;
    static KnowledgeLevel fromScore(int score) {
        if (score <= 3) return BEGINNER;
        else if (score <= 6) return ADVANCED;
        else if (score <= 9) return PROFESSIONAL;
        else return MASTER;
    }
}

public class KnowledgeDemo {
    public static void main(String[] args) {
        System.out.println(KnowledgeLevel.fromScore(0));
        System.out.println(KnowledgeLevel.fromScore(5));
        System.out.println(KnowledgeLevel.fromScore(9));
        System.out.println(KnowledgeLevel.fromScore(10));
    }
}
```

Sample Output

```
BEGINNER
ADVANCED
PROFESSIONAL
MASTER
```

---

14. Write a Program for Exception Handling – Division & Array Access

```
public class ExceptionDemo {
    public static void main(String[] args) {
        try {
            int a = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Division by zero is not allowed!");
        } finally {
            System.out.println("Operation completed.");
        }

        try {
            int[] arr = new int[3];
            int val = arr[5];
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index out of bounds!");
        } finally {
            System.out.println("Operation completed.");
        }
    }
}
```

Sample Output

```
Division by zero is not allowed!
Operation completed.
Array index out of bounds!
Operation completed.
```

---

15. Write a Program for Exception Handling – Custom Exception (Odd Number)

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

class OddChecker {
    public static void checkOdd(int n) throws OddNumberException {
        if (n % 2 != 0) throw new OddNumberException("Odd number: " + n);
        else System.out.println("Even number: " + n);
    }
}

public class OddCheckDemo {
    public static void main(String[] args) {
        try {
            OddChecker.checkOdd(3);
        } catch (OddNumberException e) {
            System.out.println(e.getMessage());
        }
        try {
            OddChecker.checkOdd(4);
        } catch (OddNumberException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Sample Output

```
Odd number: 3
Even number: 4
```

---

16. Write a Program for File Handling – Multiple Catches

```
import java.io.*;

public class FileReadDemo {
    public static void main(String[] args) {
        String filename = "test.txt";
        try {
            readFile(filename);
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + filename);
        } catch (IOException e) {
            System.out.println("Error reading file: " + e.getMessage());
        } finally {
            System.out.println("Cleanup done.");
        }
    }

    public static void readFile(String filename) throws FileNotFoundException, IOException {
        BufferedReader br = new BufferedReader(new FileReader(filename));
        String line = br.readLine();
        System.out.println(line);
        br.close();
    }
}
```

Sample Output (if file missing)

```
File not found: test.txt
Cleanup done.
```

Sample Output (if file exists with first line Hello)

```
Hello
Cleanup done.
```

---

17. Write a Program for Multiple Exceptions in One Try Block

```
import java.io.*;

public class MultiExceptionDemo {
    public static void main(String[] args) {
        try {
            BufferedReader br = new BufferedReader(new FileReader("numbers.txt"));
            String line = br.readLine();
            int num = Integer.parseInt(line);
            int result = 100 / num;
            System.out.println(result);
            br.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found");
        } catch (IOException e) {
            System.out.println("Problem reading file");
        } catch (NumberFormatException e) {
            System.out.println("Invalid number format");
        } catch (ArithmeticException e) {
            System.out.println("Division by zero");
        } finally {
            System.out.println("Execution completed");
        }
    }
}
```

Sample Output (file missing)

```
File not found
Execution completed
```

Sample Output (file contains abc)

```
Invalid number format
Execution completed
```

Sample Output (file contains 0)

```
Division by zero
Execution completed
```

Sample Output (file contains 25)

```
4
Execution completed
```