1. Write a program to:

- Read an int value from user input.
- Assign it to a double (implicit widening) and print both.
- Read a double, explicitly cast it to int, then to short, and print results (show truncation/overflow).

```
class TypeCastingDemo {
   public static void main(String[] args) throws java.io.IOException {
      java.io.BufferedReader br = new java.io.BufferedReader(new
java.io.InputStreamReader(System.in));

      System.out.print("Enter an integer: ");
      int i = Integer.parseInt(br.readLine());
      double d = i;
      System.out.println("int value = " + i);
      System.out.println("double value = " + d);

      System.out.print("Enter a double: ");
      double dd = Double.parseDouble(br.readLine());
      int ii = (int) dd;
      short s = (short) ii;
      System.out.println("double input = " + dd);
      System.out.println("int after cast = " + ii);
      System.out.println("short after cast = " + s);
   }
}
```

Output:

```
Enter an integer: 100
int value = 100
double value = 100.0
Enter a double: 130.75
double input = 130.75
```

int after cast = 130

short after cast = 130

2. Convert an int to String using String.valueOf(...), then back with Integer.parseInt(...). Handle NumberFormatException.

```
class StringConversion {
   public static void main(String[] args) throws java.io.IOException {
      java.io.BufferedReader br = new java.io.BufferedReader(new
java.io.InputStreamReader(System.in));
      System.out.print("Enter an integer string: ");
      String str = br.readLine();
      try {
         int num = Integer.parseInt(str);
         String s = String.valueOf(num);
         System.out.println("Integer: " + num);
         System.out.println("String after conversion: " + s);
      } catch (NumberFormatException e) {
         System.out.println("Invalid integer format");
      }
   }
}
```

Output:

Enter an integer string: 1234

Integer: 1234

String after conversion: 1234

3. Compound Assignment Behaviour

- Initialize int x=5;
- x = x + 4.5; // Does this compile?
- x += 4.5; // What happens here?
- Print results.

```
class CompoundAssignment {
```

```
    public static void main(String[] args) {
        int x = 5;
        // x = x + 4.5; // Compile error: cannot convert double to int
        x += 4.5; // Works: implicit cast after compound assignment
        System.out.println("x after x += 4.5: " + x);
    }
}
```

Output:

x after x += 4.5: 9

4. Object Casting with Inheritance

- Define Animal class with makeSound().
- Define Dog subclass overriding makeSound() and method fetch().
- In main, upcast Dog to Animal and call makeSound().

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

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

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

```
        Dog d = new Dog();
        Animal a = d;
        a.makeSound();
    }
}
```

Output:

Woof!

5. Mini Project – Temperature Converter

- Prompt user Celsius (double)
- Convert to Fahrenheit = celsius*9/5 + 32
- Cast Fahrenheit to int
- Print both values

```
class TemperatureConverter {
    public static void main(String[] args) throws java.io.IOException {
        java.io.BufferedReader br = new java.io.BufferedReader(new
java.io.InputStreamReader(System.in));
        System.out.print("Enter temperature in Celsius: ");
        double celsius = Double.parseDouble(br.readLine());
        double fahrenheit = celsius * 9 / 5 + 32;
        int fInt = (int) fahrenheit;
        System.out.println("Fahrenheit (double): " + fahrenheit);
        System.out.println("Fahrenheit (int): " + fInt);
    }
}
```

Output:

Enter temperature in Celsius: 36.6

Fahrenheit (double): 97.88

Fahrenheit (int): 97

6. Enum: Days of the Week

Define enum DaysOfWeek

- Prompt user to input a day
- Print ordinal and check if weekend

```
enum DaysOfWeek { Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday }


class DaysDemo {
   public static void main(String[] args) throws java.io.IOException {
      java.io.BufferedReader br = new java.io.BufferedReader(new
java.io.InputStreamReader(System.in));
      System.out.print("Enter day: ");
      String dayStr = br.readLine();
      try {
         DaysOfWeek day = DaysOfWeek.valueOf(dayStr);
         System.out.println("Position: " + day.ordinal());
         if (day == DaysOfWeek.Saturday || day == DaysOfWeek.Sunday)
            System.out.println(day + " is weekend");
         else
            System.out.println(day + " is weekday");
      } catch (IllegalArgumentException e) {
         System.out.println("Invalid day");
      }
   }
}
```

Output:

Enter day: Sunday

Position: 6

Sunday is weekend

7. Enum: Compass Directions

- Define Direction enum
- Read Direction from string
- Print move message

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


class DirectionDemo {
   public static void main(String[] args) throws java.io.IOException {
      java.io.BufferedReader br = new java.io.BufferedReader(new
java.io.InputStreamReader(System.in));
      System.out.print("Enter direction: ");
      String dirStr = br.readLine();
      try {
         Direction dir = Direction.valueOf(dirStr);
         switch(dir) {
            case NORTH: System.out.println("Move north"); break;
            case SOUTH: System.out.println("Move south"); break;
            case EAST: System.out.println("Move east"); break;
            case WEST: System.out.println("Move west"); break;
         }
      } catch (IllegalArgumentException e) {
         System.out.println("Invalid direction");
      }
   }
}
```

Output:

```
Enter direction: EAST
Move east
```

8. Enum: Shape Area Calculator

```
enum Shape {
   CIRCLE {
      double area(double... params) {
         return 3.14 * 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);
}

class ShapeDemo {
    public static void main(String[] args) {
        System.out.println("Circle area: " + Shape.CIRCLE.area(5));
        System.out.println("Square area: " + Shape.SQUARE.area(4));
        System.out.println("Rectangle area: " + Shape.RECTANGLE.area(3,6));
        System.out.println("Triangle area: " + Shape.TRIANGLE.area(3,6));
    }
}
```

Output:

Circle area: 78.5

Square area: 16.0

Rectangle area: 18.0

Triangle area: 9.0

9. Card Suit & Rank

- Redesign a Card class using enums Suit and Rank
- Create Deck with all 52 cards
- Shuffle and print order

```
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 {
   Card[] cards = new Card[52];
   Deck() {
      int index = 0;
      for (Suit s : Suit.values()) {
         for (Rank r : Rank.values()) {
            cards[index++] = new Card(s, r);
         }
      }
```

```
    }
    void shuffle() {
        for (int i = 0; i < cards.length; i++) {
            int j = (int)(Math.random() * cards.length);
            Card temp = cards[i];
            cards[i] = cards[j];
            cards[j] = temp;
        }
    }
    void printDeck() {
        for (Card c : cards) {
            System.out.println(c);
        }
    }
}

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

Output:

QUEEN of CLUBS

SEVEN of SPADES

THREE of DIAMONDS

... (rest shuffled)

10. Priority Levels with Extra Data

```
enum PriorityLevel {
```

```
    LOW(1), MEDIUM(3), HIGH(5), CRITICAL(7);
    int severity;
    PriorityLevel(int s) { severity = s; }
    boolean isUrgent() { return severity >= 5; }
}

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

Output:

```
LOW severity: 1, urgent? false
MEDIUM severity: 3, urgent? false
HIGH severity: 5, urgent? true
CRITICAL severity: 7, urgent? true
```

11. 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; }
    };
}

class TrafficDemo {
    public static void main(String[] args) {
        State state = TrafficLight.RED;
        for (int i = 0; i < 6; i++) {
            System.out.println(state);
            state = state.next();
        }
    }
}
```

Output:

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

12. Difficulty Level & Game Setup

```
enum Difficulty { EASY, MEDIUM, HARD }

class Game {
    Game(Difficulty diff) {
        int bullets = 0;
        switch(diff) {
            case EASY: bullets = 3000; break;
            case MEDIUM: bullets = 2000; break;
```

```
            case HARD: bullets = 1000; break;
        }
        System.out.println(diff + " level bullets: " + bullets);
    }
}

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

Output:

EASY level bullets: 3000

MEDIUM level bullets: 2000

HARD level bullets: 1000

13. Calculator Operations Enum

```
enum Operation {
    PLUS, MINUS, TIMES, DIVIDE;

    double eval(double a, double b) {
        switch(this) {
            case PLUS: return a + b;
            case MINUS: return a - b;
            case TIMES: return a * b;
            case DIVIDE: return a / b;
        }
        return 0;
    }
```

```
}

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

Output:

8.0

2.0

15.0

1.6666666666666667

14. Exception Handling - Division & Array Access

```
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];
         System.out.println(arr[5]);
      } catch (ArrayIndexOutOfBoundsException e) {
```

```
            System.out.println("Array index out of bounds!");
        } finally {
            System.out.println("Operation completed.");
        }
    }
}
```

Output:

Division by zero is not allowed!

Operation completed.

Array index out of bounds!

Operation completed.

15. Throw and Handle Custom Exception

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

class OddChecker {
    static void checkOdd(int n) throws OddNumberException {
        if (n % 2 != 0) throw new OddNumberException("Odd number: " + n);
    }

    public static void main(String[] args) {
        int[] nums = {2, 3, 4, 5};
        for (int n : nums) {
            try {
                checkOdd(n);
                System.out.println(n + " is even");
            } catch (OddNumberException e) {
```

```
            System.out.println(e.getMessage());
        }
    }
  }
}
```

Output:

2 is even

Odd number: 3

4 is even

Odd number: 5

16. File Handling with Multiple Catches (Using mam's approach - simulate file reading with try-catch)

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

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

```
    }
}
```

Output (if file not found):

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

17. Multi Exception in One Try Block

```
class MultiExceptionDemo {
    public static void main(String[] args) {
        try {
            java.io.BufferedReader br = new java.io.BufferedReader(new java.io.FileReader("test.txt"));
            String line = br.readLine();
            int num = Integer.parseInt(line);
            System.out.println(100 / num);
            br.close();
        } catch (java.io.FileNotFoundException e) {
            System.out.println("File not found");
        } catch (java.io.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");
        }
    }
}
```

Output (example if file missing):

```
File not found
Execution completed
```