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—demonstrate truncation or overflow

Program :

```
package Day6;

import java.util.Scanner;

public class TypeCasting {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      System.out.print("Enter an int value: ");

      int num = sc.nextInt();

      double num1 = num;

      System.out.println("Int value: " + num);

      System.out.println("Widened to double: " + num1);

      System.out.print("Enter a double value: ");

      double num2 = sc.nextDouble();

      int num3 = (int) num2;

      short num4 = (short) num3;

      System.out.println("double value: " + num2);

      System.out.println("Casted to int : " + num3);

      System.out.println("Casted to short: " + num4);

   }

}
```

Output :

```
Enter an int value: 12

Int value: 12

Widened to double: 12.0

Enter a double value: 23.5

double value: 23.5
```

Casted to int : 23

Casted to short: 23

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

Program :

```
package Day6;

import java.util.Scanner;

public class String_Conv {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        try {

            System.out.print("Enter an number: ");

            int num = sc.nextInt();


            String str = String.valueOf(num);

            System.out.println("Converted to String: " + str);


            int parsed = Integer.parseInt(str);

            System.out.println("Converted back to int: " + parsed);


        } catch (NumberFormatException e) {

            System.out.println("Invalid number format!");

        }

    }

}
```

Output : Enter an number: 23

Converted to String: 23

Converted back to int: 23

3. Compound Assignment Behaviour

1. Initialize int x = 5;.

2. Write two operations:

x = x + 4.5; // Does this compile? Why or why not?

x += 4.5; // What happens here?

Print results and explain behavior in comments ?

Program :

```
package Day6;
public class Test {
    public static void main(String[] args) {
        int x = 5;
        // x = x + 4.5; // Compile error: possible lossy conversion from double to int
        x += 4.5; // Implicit narrowing after addition — works
        System.out.println(x);
    }
}
```

Ouptut : 9

4. Object Casting with Inheritance ?

Program :

```
package Day6;
class Animal {
    void makeSound() {
        System.out.println("animal makes sounds");
    }
}
class Dog extends Animal {
    void makeSound() {
        System.out.println("Bark!");
    }
    void eat() {
        System.out.println("Dog eats Royal canin.");
    }
}
```

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

Output: Bark!

5. Mini Project – Temperature Converter

Program :

```
package Day6;
import java.util.Scanner;
public class Temp_Converter {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter temperature in Celsius: ");
        double celsius = sc.nextDouble();

        double fahrenheit = celsius * 9 / 5 + 32;
        int fahrenheit1 = (int) fahrenheit;
        System.out.println("Fahrenheit actual value: " + fahrenheit);
        System.out.println("Fahrenheit in integer: " + fahrenheit1);
    }
}
```

Output : Enter temperature in Celsius: 37

Fahrenheit actual value: 98.6

Fahrenheit in integer: 98

Enum :

1. Days of the Week ?

Program :

```
package Day6;
import java.util.Scanner;
enum DaysOfWeek {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
}
public class enumDay {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a day name: ");
        String input = sc.next().toUpperCase();
            DaysOfWeek day = DaysOfWeek.valueOf(input);
            System.out.println("Position: " + day.ordinal());
            switch (day) {
                case SATURDAY:
                case SUNDAY:
                    System.out.println(day + " is a weekend.");
                    break;
                default:
                    System.out.println(day + " is a weekday.");
            }
        }
    }
```

Output : Enter a day name: SUNDAY

Position: 6

SUNDAY is a weekend.

2. Compass Directions ?

Program :

```
package Day6;
import java.util.Scanner;
enum Direction {
```

```
    NORTH, SOUTH, EAST, WEST
}
public class Direction {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a direction (NORTH, SOUTH, EAST, WEST): ");
        String input = sc.next().toUpperCase();
            Direction dir = Direction.valueOf(input);
            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;
            }
    }
}
```

Output : Enter a direction (NORTH, SOUTH, EAST, WEST): EAST

Move east

3. Shape Area Calculator ?

Program :

```
package Day6;
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 Test {
    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(4, 6));
        System.out.println("Triangle area: " + Shape.TRIANGLE.area(4, 5));
    }
}
```

Output : Circle area: 78.53981633974483

Square area: 16.0

Rectangle area: 24.0

Triangle area: 10.0

4. Card Suit and Rank ?

Program :

```
package Day6;
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 suit, Rank rank) {
      this.suit = suit;
      this.rank = rank;
   }
   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 Cards {
    public static void main(String[] args) {
        Deck deck = new Deck();
        deck.shuffle();
        deck.printDeck();
    }
}
```

Output : ACE of SPADES

KING of DIAMONDS

KING of HEARTS

TWO of DIAMONDS

JACK of CLUBS

QUEEN of DIAMONDS

EIGHT of CLUBS

ACE of HEARTS

THREE of CLUBS

TEN of SPADES

SEVEN of SPADES

TWO of HEARTS

SEVEN of CLUBS

TEN of DIAMONDS

SEVEN of DIAMONDS

ACE of DIAMONDS

THREE of DIAMONDS

NINE of CLUBS

JACK of DIAMONDS

KING of SPADES

QUEEN of SPADES

NINE of HEARTS

EIGHT of DIAMONDS

TEN of CLUBS

FIVE of DIAMONDS

KING of CLUBS

QUEEN of HEARTS

FIVE of CLUBS

FOUR of CLUBS

EIGHT of SPADES

ACE of CLUBS

SIX of CLUBS

JACK of SPADES

SIX of SPADES

TWO of SPADES

THREE of SPADES

EIGHT of HEARTS

QUEEN of CLUBS

FIVE of SPADES

FOUR of HEARTS

FOUR of DIAMONDS

FIVE of HEARTS

JACK of HEARTS

NINE of SPADES

TEN of HEARTS

NINE of DIAMONDS

SIX of DIAMONDS

FOUR of SPADES

SIX of HEARTS

SEVEN of HEARTS

TWO of CLUBS

THREE of HEARTS

5. Priority levels with extra data ?

Program :

```
package Day6;
enum PriorityLevel {
   LOW(1), MEDIUM(2), HIGH(3), CRITICAL(4);
   private int severity;
   PriorityLevel(int severity) {
      this.severity = severity;
   }
   public boolean isUrgent() {
      return severity >= 3;
   }
   public int getSeverity() {
      return severity;
   }
}
public class Priority {
   public static void main(String[] args) {
      for (PriorityLevel p : PriorityLevel.values()) {
         System.out.println(p + " (Severity: " + p.getSeverity() +
                        ", Urgent: " + p.isUrgent() + ")");
      }
```

```
    }
}
```

Output : LOW (Severity: 1, Urgent: false)

MEDIUM (Severity: 2, Urgent: false)

HIGH (Severity: 3, Urgent: true)

CRITICAL (Severity: 4, Urgent: true)

6. Traffic Light State Machine ?

Program :

```
package Day6;
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 Traffic_light {
    public static void main(String[] args) {
        State current = TrafficLight.RED;
        for (int i = 0; i < 6; i++) {
            System.out.println("Light: " + current);
            current = current.next();
        }
    }
}
```

Output : Light: RED

Light: GREEN

Light: YELLOW

Light: RED

Light: GREEN

Light: YELLOW

7. Difficulty Level and Game Setup ?

Program :

```
package Day6;
enum Difficulty {
    EASY, MEDIUM, HARD
}
class Game {
    Game(Difficulty diff) {
        switch (diff) {
            case EASY:
                System.out.println("3000 bullets");
                break;
            case MEDIUM:
                System.out.println("2000 bullets");
                break;
```

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

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

Output : 2000 bullets

8. Calculator Operations using Enum ?

Program :

```
package Day6;
public class Calculator {
 enum OperationSwitch {
    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;
        }
```

```
    }
}
enum OperationOverride {
    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 static void main(String[] args) {
    System.out.println(OperationSwitch.PLUS.eval(5, 3));
    System.out.println(OperationOverride.TIMES.eval(5, 3));
}
}
```

Output : 8.0

15.0

9. Knowledge Level from Score Range?

Program :

```
package Day6;
public class Score {
 enum KnowledgeLevel {
    BEGINNER,
    ADVANCED,
    PROFESSIONAL,
    MASTER;
    static KnowledgeLevel fromScore(int score) {
       if (score >= 0 && score <= 3) {
          return BEGINNER;
       } else if (score <= 6) {
          return ADVANCED;
       } else if (score <= 9) {
          return PROFESSIONAL;
       } else if (score == 10) {
          return MASTER;
       }
    }
 }
 public static void main(String[] args) {
    int[] testScores = {0, 3, 4, 6, 7, 9, 10};
    for (int score : testScores) {
       System.out.println(score + " - " + KnowledgeLevel.fromScore(score));
    }
 }
}
```

Output : 0 - BEGINNER

3 - BEGINNER

4 - ADVANCED

6 - ADVANCED

7 - PROFESSIONAL

9 - PROFESSIONAL

10 - MASTER

Exception Handling :

1. Division & Array Access ?

Program :

```
package Day6;
public class Test {
   public static void main(String[] args) {
      // Division by zero
      try {
         int result = 10 / 0;
         System.out.println("Result: " + result);
      } catch (ArithmeticException e) {
         System.out.println("Division by zero is not allowed!");
      } finally {
         System.out.println("Operation completed.");
      }
      // Array index out of bounds
      try {
         int[] arr = {1, 2, 3};
         System.out.println(arr[5]);
      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Invalid array index!");
      } finally {
         System.out.println("Operation completed.");
      }
```

```
    }
}
```

Output : Division by zero is not allowed!

Operation completed.

Invalid array index!

Operation completed.

2. Throw and Handle Custom Exception.

Program :

```
package Day6;
class OddNumberException extends Exception {
    public OddNumberException(String message) {
        super(message);
    }
}
public class Custom_exception {
    public static void checkOdd(int n) throws OddNumberException {
        if (n % 2 != 0) {
            throw new OddNumberException("Odd number: " + n);
        } else {
            System.out.println(n + " is even.");
        }
    }
    public static void main(String[] args) {
        int[] numbers = {2, 5, 8};
        for (int num : numbers) {
            try {
                checkOdd(num);
            } catch (OddNumberException e) {
                System.out.println(e );
            }
```

```
        }
    }
}
```

Output : 2 is even.

Day6.OddNumberException: Odd number: 5

8 is even.

3.File Handling with multiple catches

Program :

```
package Day6;
import java.io.*;
public class File_Handling {
    public static void readFile(String filename) throws FileNotFoundException, IOException {
        BufferedReader br = new BufferedReader(new FileReader(filename));
        String line = br.readLine();
        System.out.println("First line: " + line);
        br.close();
    }
    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 );
        } finally {
            System.out.println("Cleanup done.");
        }
    }
}
```

Output : File not found: test.txt

Cleanup done.

4. Multi Exception in one Try block.

Program :

```
package Day6;
import java.io.*;
public class Multi_Exception {
    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: " + 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");
        }
    }
}
```

Output : File not found.

Execution completed