7. Java Enumeration

Enumeration (enum) is a special data type introduced in Java 1.5 to define a collection of constants. It is used to define variables that can only take one out of a predefined set of values.

Key Features:

- Strongly typed constants.
- Provides type safety.
- Can include methods and fields.

Example Program:

```
enum Day {
   MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY;
}
public class EnumExample {
   public static void main(String[ ] args) {
      Day today = Day.MONDAY;

      System.out.println("Today is: " + today); // Print the value

      // Loop through all enum values
      System.out.println("Days of the week:");
      for (Day day : Day.values( ))
         System.out.println(day);
   }     }
```

Output:

```
Today is: MONDAY
Days of the week:
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY
```

8. Class Math

The Math class provides methods for performing basic numeric operations like trigonometric, logarithmic, and exponential calculations.

Common Methods:

- Math.abs( ): Absolute value.
- Math.sqrt( ): Square root.
- Math.pow( ): Power.
- Math.random( ): Generates a random number between 0.0 and 1.0.

Example Program:

```
public class MathExample {
   public static void main(String[ ] args) {
      double num = -25.5;

      System.out.println("Absolute value: " + Math.abs(num));
      System.out.println("Square root: " + Math.sqrt(16));
      System.out.println("Power (2^3): " + Math.pow(2, 3));
      System.out.println("Random number: " + Math.random( ));
   }     }
```

Output:

```
Absolute value: 25.5
Square root: 4.0
Power (2^3): 8.0
Random number: (varies)
```

9. Wrapper Classes

Wrapper classes in Java provide a way to use primitive data types as objects. Examples include Integer, Double, Character, and Boolean.

Purpose:

- Required for collections like ArrayList.
- Useful for converting between primitives and objects.

Example Program:

```
public class WrapperExample {
   public static void main(String[ ] args) {
      int num = 10;

      Integer wrappedNum = Integer.valueOf(num); // Boxing
      int unwrappedNum = wrappedNum.intValue( ); // Unboxing

      System.out.println("Wrapped value: " + wrappedNum);
      System.out.println("Unwrapped value: " + unwrappedNum);
   }     }
```

Output:

```
Wrapped value: 10
Unwrapped value: 10
```

10. Auto-boxing and Auto-unboxing

Auto-boxing and auto-unboxing simplify the process of converting between primitive types and their corresponding wrapper classes.

- Auto-boxing: Primitive → Wrapper
- Auto-unboxing: Wrapper → Primitive

Example Program:

```
import java.util.ArrayList;

public class AutoBoxingExample {
   public static void main(String[ ] args) {
      ArrayList<Integer> numbers = new ArrayList<>( );

      // Auto-boxing
      numbers.add(5); // primitive is automatically converted to Integer

      // Auto-unboxing
      int num = numbers.get(0); // Integer is automatically converted to int

      System.out.println("Stored number: " + num);
   }
}
```

Output:

```
Stored number: 5
```

11. Temporal Adjusters Class

TemporalAdjusters is a utility class in the java.time package that adjusts a Temporal object (e.g., LocalDate) to a different date based on certain rules.

Common Adjusters:

- firstDayOfMonth( )
- lastDayOfMonth( )
- next( ) and previous( )

Example Program:

```
import java.time.LocalDate;
import java.time.temporal.TemporalAdjusters;

public class TemporalAdjusterExample {
   public static void main(String[ ] args) {
      LocalDate today = LocalDate.now( );

      // First day of the month
      LocalDate firstDay = today.with(TemporalAdjusters.firstDayOfMonth( ));

      // Last day of the month
      LocalDate lastDay = today.with(TemporalAdjusters.lastDayOfMonth( ));

      // Next Sunday
      LocalDate nextSunday = today.with(TemporalAdjusters.next(java.time.DayOfWeek.SUNDAY));

      System.out.println("Today's date: " + today);
      System.out.println("First day of the month: " + firstDay);
      System.out.println("Last day of the month: " + lastDay);
      System.out.println("Next Sunday: " + nextSunday);
   }
}
```

Output:

```
Today's date: 2024-11-07
First day of the month: 2024-11-01
Last day of the month: 2024-11-30
Next Sunday: 2024-11-10
```