Wrapper Classes

Wrapper classes are a set of classes that allow you to convert primitive data types into objects and vice versa. They are used to wrap primitive data types, such as int, double, char, etc., inside an object. With the help of these wrapper classes, primitive data types can be treated as objects. This becomes useful if we want use these data types in collections or passing them as parameters to methods that expect objects.

The eight wrapper classes in Java correspond to the eight primitive data types:

- Byte: Wraps a byte primitive type.
- Short: Wraps a short primitive type.
- Integer: Wraps an int primitive type.
- Long: Wraps a long primitive type.
- Float: Wraps a float primitive type.
- Double: Wraps a double primitive type.
- Character: Wraps a char primitive type.
- Boolean: Wraps a boolean primitive type.

Wrapper classes are part of the java.lang package.They are automatically imported into every Java program. They provide useful methods for converting between primitive types and objects and performing various operations on the wrapped values.

Arrays Wrapper Class

It is a wrapper class present java.util package. It provides various utility methods to work with arrays in java. These methods are used to simplify common array-related tasks and it is very useful for array manipulation. It includes methods for sorting arrays, searching for elements, filling arrays with specific values, and converting arrays to strings.

The methods present in Arrays wrapper class are:

1. Arrays.sort()

This method is used to sort the elements of an array in ascending order.

Example:

```
import java.util.Arrays;

public class ArraySortExample {
   public static void main(String[] args) {
      int[] numbers = {5, 2, 8, 1, 3};
```

```
        Arrays.sort(numbers);

        System.out.println(Arrays.toString(numbers));
    }
}
```

2. Arrays.binarySearch()

This method searches for a specified element in a sorted array using binary search and returns its index if found otherwise, it returns a negative value.

Example:

```
import java.util.Arrays;

public class ArrayBinarySearchExample {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 5, 8};

        int index = Arrays.binarySearch(numbers, 3);

        System.out.println("Index of 3: " + index);
    }
}
```

3. Arrays.copyOf()

This method creates a copy of an array with a specified length.

Example:

```
import java.util.Arrays;

public class ArrayCopyExample {
    public static void main(String[] args) {
        int[] sourceArray = {1, 2, 3, 4, 5};

        int[] targetArray = Arrays.copyOf(sourceArray, 3);

        System.out.println(Arrays.toString(targetArray));
    }
}
```

4. Arrays.fill()

This method fills the elements of an array with a specified value.

Example:

```
import java.util.Arrays;

public class ArrayFillExample {
    public static void main(String[] args) {
        int[] numbers = new int[5];

        Arrays.fill(numbers, 10);

        System.out.println(Arrays.toString(numbers));
    }
}
```

5. Arrays.equals()

This method checks if two arrays are equal, i.e., they have the same length and contain the same elements in the same order.

Example:

```
import java.util.Arrays;

public class ArrayEqualsExample {
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3};
        int[] arr2 = {1, 2, 3};

        boolean isEqual = Arrays.equals(arr1, arr2);

        System.out.println("Are arrays equal? " + isEqual);
    }
}
```

Math Class

The Math class is a built-in utility class provided by Java's standard library, and it contains various static methods for performing common mathematical operations.

The common methods present in the Math class are:

1. Math.sqrt()

This method returns the square root of a given number.

Example:

```
public class SquareRootExample {
```

```
    public static void main(String[] args) {
    double number = 16.0;

    double squareRoot = Math.sqrt(number);
    System.out.println("Square root of " + number + " is " + squareRoot);
  }
}
```

2. Math.pow()

This method is used to calculate a number raise to the power of some other number.

Example:

```
public class PowerExample {
  public static void main(String[] args) {
    double base = 2.0;
    double exponent = 3.0;

    double result = Math.pow(base, exponent);

    System.out.println(base + " raised to the power of " + exponent + " is " + result);
  }
}
```