ASSIGNMENT NO.2

Note: Consider the following before starting the assignment:

- A **static field** declared inside a class is called a **class-level variable**. To access this variable, use the class name and the dot operator (e.g., `Integer.MAX_VALUE`).
- A **static method** defined inside a class is called a **class-level method**. To access this method, use the class name and the dot operator (e.g., `Integer.parseInt()`).
- When accessing static members within the same class, you do not need to use the class name.

1. Working with `java.lang.Boolean`

a. Explore the Java API documentation for `java.lang.Boolean` and observe its modifiers and super types.

b. Declare a method-local variable `status` of type `boolean` with the value `true` and convert it to a `String` using the `toString` method. (Hint: Use `Boolean.toString(Boolean)`).

```
package org.cdac;



public class BooleanExample {

  public static void main(String[] args) {

    boolean status = true;

    String strStatus = Boolean.toString(status);

    System.out.println("Boolean as String: " + strStatus);

  }

}
```

```
<terminated> BooleanExample [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.open
Boolean as String: true
```

c. Declare a method-local variable `strStatus` of type `String` with the value `"true"` and convert it to a `boolean` using the `parseBoolean` method. (Hint: Use `Boolean.parseBoolean(String)`).

```
package org.cdac;


public class BooleanExample2 {

  public static void main(String[] args) {

    String strStatus = "true";

    boolean status = Boolean.parseBoolean(strStatus);

    System.out.println("String to boolean: " + status);

  }

}
```

```
<terminated> BooleanExample2 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
String to boolean: true
```

d. Declare a method-local variable `strStatus` of type `String` with the value `"1"` or `"0"` and attempt to convert it to a `boolean`. (Hint: `parseBoolean` method will not work as expected with `"1"` or `"0"`).

```
package org.cdac;

public class BooleanExample4 {

  public static void main(String[] args) {

    String strStatus = "1";

    boolean status = Boolean.parseBoolean(strStatus); // Will return false

    System.out.println("String '1' to boolean: " + status);

  }

}
```

```
<terminated> BooleanExample4 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240
String '1' to boolean: false
```

e. Declare a method-local variable `status` of type `boolean` with the value `true` and convert it to the corresponding wrapper class using `Boolean.valueOf()`. (Hint: Use `Boolean.valueOf(boolean)`).

```
package org.cdac;

public class BooleanExample5 {

   public static void main(String[] args) {

      boolean status = true;

      Boolean wrapperStatus = Boolean.valueOf(status);

      System.out.println("Boolean wrapper class: " + wrapperStatus);

   }

}
```

```
<terminated> BooleanExample5 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v202
Boolean wrapper class: true
```

f. Declare a method-local variable `strStatus` of type `String` with the value `"true"` and convert it to the corresponding wrapper class using `Boolean.valueOf()`. (Hint: Use `Boolean.valueOf(String)`).

```
package org.cdac;

public class BooleanExample6 {

   public static void main(String[] args) {

      String strStatus = "true";

      Boolean wrapperStatus = Boolean.valueOf(strStatus);

      System.out.println("String to Boolean wrapper class: " + wrapperStatus);

   }

}
```

```
<terminated> BooleanExample6 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu
String to Boolean wrapper class: true
```

g. Experiment with converting a `boolean` value into other primitive types or vice versa and observe the results.

```
package org.cdac;


public class BooleanExample7 {

    public static void main(String[] args) {

        boolean status = true;

        int intValue = status ? 1 : 0;

        System.out.println("Boolean to int: " + intValue);

        String stringValue = Boolean.toString(status);

        System.out.println("Boolean to String: " + stringValue);

        boolean fromString = Boolean.parseBoolean("true");

        System.out.println("String to boolean: " + fromString);

    }

}
```

```
Console ×
<terminated> BooleanExample7 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.
Boolean to int: 1
Boolean to String: true
String to boolean: true
```

2. Working with `java.lang.Byte`

a. Explore the Java API documentation for `java.lang.Byte` and observe its modifiers and super types.

b. Write a program to test how many bytes are used to represent a `byte` value using the `BYTES` field. (Hint: Use `Byte.BYTES`).

```
package org.cdac;

public class ByteExample {

	public static void main (String args[]) {

		System.out.println("Bytes used to repesent a byte:"+Byte.BYTES);

	}

}
```

```
Bytes used to repesent a byte:1
```

c. Write a program to find the minimum and maximum values of `byte` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Byte.MIN_VALUE` and `Byte.MAX_VALUE`).

```
package org.cdac;

public class ByteExample2 {

	public static void main (String args[]) {

		System.out.println("Mininum byte value:"+Byte.MIN_VALUE);

		System.out.println("Maximum byte value:"+Byte.MAX_VALUE);

	}

}
```

```
<terminated> ByteExample2 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\
Mininum byte value:-128
Maximum byte value:127
```

d. Declare a method-local variable `number` of type `byte` with some value and convert it to a `String` using the `toString` method. (Hint: Use `Byte.toString(byte)`).

```
package org.cdac;

public class ByteExample3 {

	public static void main (String args[]) {

		byte number =3;

		String strnumber=Byte.toString(number);

		System.out.println("Byte as String:"+strnumber);

	}

}
```

```
Console ×
<terminated> ByteExample3 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-15
Byte as String:3
```

e. Declare a method-local variable `strNumber` of type `String` with some value and convert it to a `byte` value using the `parseByte` method. (Hint: Use `Byte.parseByte(String)`).

```
package org.cdac;

public class ByteExample4 {

	public static void main (String args[]) {

	String strNumber ="24";

	byte Number = Byte.parseByte(strNumber);

	System.out.println("String as byte:"+Number);

	}

}
```

```
String as byte:24
```

f. Declare a method-local variable `strNumber` of type `String` with the value `"Ab12Cd3"` and attempt to convert it to a `byte` value. (Hint: `parseByte` method will throw a `NumberFormatException`).

```
package org.cdac;

 class ByteExample5 {

   public static void main (String args[]) {

      String strNumber = "Ab12cd3";

      try {

         byte number = Byte.parseByte(strNumber);

         System.out.println("String to byte: " + number);

      } catch (NumberFormatException e) {

                  System.out.println("NumberFormatException: Invalid byte value in
the string '" + strNumber + "'");

      }

   }

}
```

```
Console ×
<terminated> ByteExample5 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v2
NumberFormatException: Invalid byte value in the string 'Ab12cd3'
```

g. Declare a method-local variable `number` of type `byte` with some value and convert it to the corresponding wrapper class using `Byte.valueOf()`. (Hint: Use `Byte.valueOf(byte)`).

```
package org.cdac;

class ByteExample6 {

  public static void main (String args[]) {

    byte Number = 24;

    Byte wrapperNumber = Byte.valueOf(Number);

    System.out.println("Byte wrapper class:"+wrapperNumber);

  }

}
```

```
Console ×
<terminated> ByteExample6 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (
Byte wrapper class:24
```

h. Declare a method-local variable `strNumber` of type `String` with some `byte` value and convert it to the corresponding wrapper class using `Byte.valueOf()`. (Hint: Use `Byte.valueOf(String)`).

```
package org.cdac;

class ByteExample7 {

  public static void main (String args[]) {

    String strNumber = "34";

    byte wrapperNumber = Byte.valueOf(strNumber);

    System.out.println("string to Byte wrapper class"+wrapperNumber);

  }

}
```

```
<terminated> ByteExample7 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v2024042
string to Byte wrapper class34
```

i. Experiment with converting a `byte` value into other primitive types or vice versa and observe the results.

```
package org.cdac;

public class ByteExample8{

        public static void main(String args[]) {

                byte number =10;

                int intvalue = number;

                System.out.println("bute into int :"+intvalue);

                double doublevalue = number;

                System.out.println("Byte to double: " +doublevalue);

        }

}
```

```
<terminated> ByteExample8 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspo
bute into int :10
Byte to double: 10.0
```

3. Working with `java.lang.Short`

a. Explore the Java API documentation for `java.lang.Short` and observe its modifiers and super types.

b. Write a program to test how many bytes are used to represent a `short` value using the `BYTES` field. (Hint: Use `Short.BYTES`).

```
package org.cdac;

public class shortExample {

        public static void main(String[] args) {

    System.out.println("Bytes used to show a short: " + Short.BYTES);

  }

}
```

```
<terminated> shortExample [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotsp
Bytes used to show a short: 2
```

c. Write a program to find the minimum and maximum values of `short` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Short.MIN_VALUE` and `Short.MAX_VALUE`).

```
package org.cdac;

public class shortExample2 {

        public static void main(String[] args) {

    System.out.println("Minimum short value: " + Short.MIN_VALUE);

    System.out.println("Maximum short value: " + Short.MAX_VALUE);

  }

}
```

```
<terminated> shortExample2 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspo
Minimum short value: -32768
Maximum short value: 32767
```

d. Declare a method-local variable `number` of type `short` with some value and convert it to a `String` using the `toString` method. (Hint: Use `Short.toString(short)`).

```
package org.cdac;

public class shortExample3 {

        public static void main(String[] args) {

    short number = 123;

    String strNumber = Short.toString(number);

    System.out.println("Short as String: " + strNumber);}

}
```

```
<terminated> shortExample3 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotsp
Short as String: 123
```

e. Declare a method-local variable `strNumber` of type `String` with some value and convert it to a `short` value using the `parseShort` method. (Hint: Use `Short.parseShort(String)`).

```
package org.cdac;

public class shortExample4 {

        public static void main(String[] args) {

    String strNumber = "346";

    short number = Short.parseShort(strNumber);

    System.out.println("String to short: " + number);

  }

}
```

```
terminated> shortExample4 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspo
String to short: 346
```

f. Declare a method-local variable `strNumber` of type `String` with the value `"Ab12Cd3"` and attempt to convert it to a `short` value. (Hint: `parseShort` method will throw a `NumberFormatException`).

```
package org.cdac;

public class shortExample5 {

        public static void main(String[] args) {

    String strNumber = "Ab12Cd3";

    try {

      short number = Short.parseShort(strNumber);
```

```
            System.out.println("String to short: " + number);

        } catch (NumberFormatException e) {

            System.out.println("NumberFormatException: Invalid short value in the string
'" + strNumber + "'");

        }

    }

}
```

```
<terminated> shortExample5 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotsp
NumberFormatException: Invalid short value in the string 'Ab12Cd3'
```

g. Declare a method-local variable `number` of type `short` with some value and convert it to the corresponding wrapper class using `Short.valueOf()`. (Hint: Use `Short.valueOf(short)`).

```
package org.cdac;

public class shortExample6 {

        public static void main(String[] args) {

            short number = 666;

            Short wrapperNumber = Short.valueOf(number);

            System.out.println("Short wrapper class: " + wrapperNumber);

        }

}
```

```
<terminated> shortExample6 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspo
Short wrapper class: 666
```

h. Declare a method-local variable `strNumber` of type `String` with some `short` value and convert it to the corresponding wrapper class using `Short.valueOf()`. (Hint: Use `Short.valueOf(String)`).

```
package org.cdac;

public class shortExample7 {

        public static void main(String[] args) {

    String strNumber = "212";

    Short wrapperNumber = Short.valueOf(strNumber);

    System.out.println("String to Short wrapper class: " + wrapperNumber);

  }

}
```

```
Console
<terminated> shortExample7 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspo
String to Short wrapper class: 212
```

i. Experiment with converting a `short` value into other primitive types or vice versa and observe the results.

```
 package org.cdac;

public class shortExample8 {  public static void main(String[] args) {

 short number = 043;

  int intValue = number;

  System.out.println("Short to int: " + intValue);

  long longValue = number;

  System.out.println("Short to long: " + longValue);

  double doubleValue = number;

  System.out.println("Short to double: " + doubleValue);
```

```
    int anotherNumber = 32000;

    short shortValue = (short) anotherNumber;

    System.out.println("Int to short (with casting): " + shortValue);

}

}
```

```
<terminated> shortExample8 [Java Application] C:\eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotsp
Short to int: 35
Short to long: 35
Short to double: 35.0
Int to short (with casting): 32000
```