1) Define a thread. Explain the two ways of creating threads with example.

Ans:- In Java, a **thread** is the smallest unit of a process that can be executed independently. Threads allow concurrent execution of tasks, enabling programs to perform multiple operations simultaneously. Java provides two ways to create threads: by extending the `Thread` class and by implementing the `Runnable` interface.

1. Extending the Thread class:

You can create a thread by extending the `Thread` class and overriding its `run()` method, which contains the code to be executed by the thread. Here's an example:

```
class MyThread extends Thread {
  public void run() {
    for (int i = 1; i <= 5; i++) {
      System.out.println("Thread 1 - Count: " + i);
    }}}
public class Main {
  public static void main(String[] args) {
    MyThread thread1 = new MyThread();
    thread1.start(); // Start the first thread

    // Main thread continues executing concurrently with thread1

    for (int i = 1; i <= 5; i++) {
      System.out.println("Main Thread - Count: " + i);
    }}}
```

2. Implementing the Runnable interface:

You can create a thread by implementing the `Runnable` interface. This approach allows you to separate the thread's behavior from the thread itself. Here's an example:

```
class MyRunnable implements Runnable {
  public void run() {
    for (int i = 1; i <= 5; i++) {
      System.out.println("Thread 2 - Count: " + i);
    }}}
public class Main {
  public static void main(String[] args) {
    Thread thread2 = new Thread(new MyRunnable());
    thread2.start(); // Start the second thread
      for (int i = 1; i <= 5; i++) {
      System.out.println("Main Thread - Count: " + i);
}}}
```

2) Explain about thread priorities in Java.

Ans:- In Java, thread priorities are used to indicate the importance or urgency of a thread. Threads with higher priorities are scheduled to run before threads with lower priorities. The Java Virtual Machine (JVM) uses thread priorities to determine the order in which threads should be executed. Thread priorities are integers ranging from 1 to 10, where 1 is the lowest priority and 10 is the highest priority. By default, all threads have a priority of 5.

```
class MyThread extends Thread {
   public MyThread(String name) {
      super(name);
   }
   public void run() {
      for (int i = 1; i <= 5; i++) {
         System.out.println(getName() + " - Count: " + i);
      }
   public class Main {
   public static void main(String[] args) {
      MyThread thread1 = new MyThread("Thread 1");
      MyThread thread2 = new MyThread("Thread 2");
      thread1.setPriority(Thread.MIN_PRIORITY);
      thread2.setPriority(Thread.MAX_PRIORITY);
      thread1.start();
      thread2.start();
   }
}
```

3) What is enumeration? Write a Java program to create an enumeration Day of Week with seven values SUNDAY through SATURDAY, Add a method isworkday() to the DayofWeek class that returns true if the value of which it is called is MONDAY through FRIDAY, otherwise false.

Ans:- In Java, an **enumeration** is a special data type that consists of a set of named constants. Enumerations are used to represent a fixed set of values. Each constant in the enumeration is implicitly final and static, making them constants. Enumerations make your code more readable and maintainable because they allow you to use meaningful names for variables.

```
// Enumeration for Days of the Week
enum DayOfWeek {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY;
    public boolean isWorkday() {
        return this != SATURDAY && this != SUNDAY;
    }
}
public class Main {
    public static void main(String[] args) {
        DayOfWeek today = DayOfWeek.MONDAY;
        System.out.println("Today is " + today);
        System.out.println("Is today a workday? " + today.isWorkday());
        DayOfWeek weekendDay = DayOfWeek.SATURDAY;
        System.out.println("Today is " + weekendDay);
        System.out.println("Is today a workday? " + weekendDay.isWorkday());
    }
}
```

4) What is autoboxing and auto unboxing in arithmetic expressions? Explain with example.
Ans:- **Autoboxing and Unboxing** in Java are features introduced in Java 5 that automatically convert primitive data types to their corresponding wrapper classes and vice versa.

- **Autoboxing:** It is the process of converting a primitive type to its corresponding wrapper class object. For example, converting an `int` to an `Integer`.
- **Unboxing:** It is the process of converting a wrapper class object to its corresponding primitive type. For example, converting an `Integer` object to an `int`.

```
// Autoboxing: converting int to Integer
int primitiveInt = 42;
Integer wrappedInt = primitiveInt;
System.out.println("Primitive Int: " + primitiveInt);
System.out.println("Wrapped Int: " + wrappedInt);
```

// Unboxing: converting Integer to int

Integer wrappedInt = 42;

int primitiveInt = wrappedInt; // unboxing

System.out.println("Wrapped Int: " + wrappedInt);

System.out.println("Primitive Int: " + primitiveInt);

//Autoboxing and AutoUnboxing

Integer a = 10; // autoboxing

Integer b = 20; // autoboxing

// Addition: unboxing, addition, autoboxing

Integer result = a + b; // unboxing, addition, autoboxing

System.out.println("Result: " + result);

5) Explain the following: i. Values() and ValuesOf() methods ii. Type Wrappers iii. compareTo()

Ans:- **Values():** In Java, the `values()` method is a special method provided by enums. It returns an array containing all the values of the enum in the order they are declared. For example, if you have an enum like this:

enum Days {

SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY;

}

ValueOf(): The `valueOf()` method is used to get an enum constant from its name. For example:

Days day = Days.valueOf("MONDAY");

ii. Type Wrappers:

In Java, primitive data types (like `int`, `float`, `char`, etc.) are not objects, so they cannot be used in collections (like ArrayLists or HashMaps) and some other data structures. To overcome this limitation, Java provides wrapper classes for each primitive data type. These wrapper classes allow primitive data types to be treated as objects.

For example:

- `Integer` is the wrapper class for `int`.
- `Double` is the wrapper class for `double`.
- `Character` is the wrapper class for `char`.

So, instead of using `int`, you can use `Integer`:

Integer num = 42;

iii. compareTo() Method:

In Java, the `compareTo()` method is used for comparing two objects. It is defined in the `Comparable` interface and is implemented by classes to provide natural ordering for their objects. For example, the `String` class in Java implements the `Comparable` interface, so you can compare strings using `compareTo()`.

```
String str1 = "apple";

String str2 = "orange";

int result = str1.compareTo(str2);

if (result < 0) {

   System.out.println("str1 comes before str2");

} else if (result > 0) {

   System.out.println("str2 comes before str1");

} else {

   System.out.println("str1 and str2 are equal");

}
```