Day-9

Exception Handling, Multi-threading, Executor Framework

Agenda

- Exception Handling
- Multi-threading
- Executor Framework

Exception Handling

- The **Exception Handling in Java** is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.
- Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.
- An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.
- **Error:** An Error indicates serious problem that a reasonable application should not try to catch.
  Exception: Exception indicates conditions that a reasonable application might try to catch.

- The try statement allows you to define a block of code to be tested for errors while it is being executed.

- The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

- Syntax:-

```
try {
   // Protected code
} catch (ExceptionName e1) {
   // Catch block
}
```

Multiple Catch Blocks

```
try {
   // Protected code
} catch (ExceptionType1 e1) {
   // Catch block
} catch (ExceptionType2 e2) {
   // Catch block
} catch (ExceptionType3 e3) {
   // Catch block
}
```

The Throws/Throw Keywords

- If a method does not handle a checked exception, the method must declare it using the **throws** keyword. The throws keyword appears at the end of a method's signature.

- You can throw an exception, either a newly instantiated one or an exception that you just caught, by using the **throw** keyword.

```
public void deposit(double amount) throws RemoteException {
    // Method implementation
    throw new RemoteException();
  }
```

The Finally Block

- The finally block follows a try block or a catch block. A finally block of code always executes, irrespective of occurrence of an Exception.

- Using a finally block allows you to run any clean-up-type statements that you want to execute, no matter what happens in the protected code.

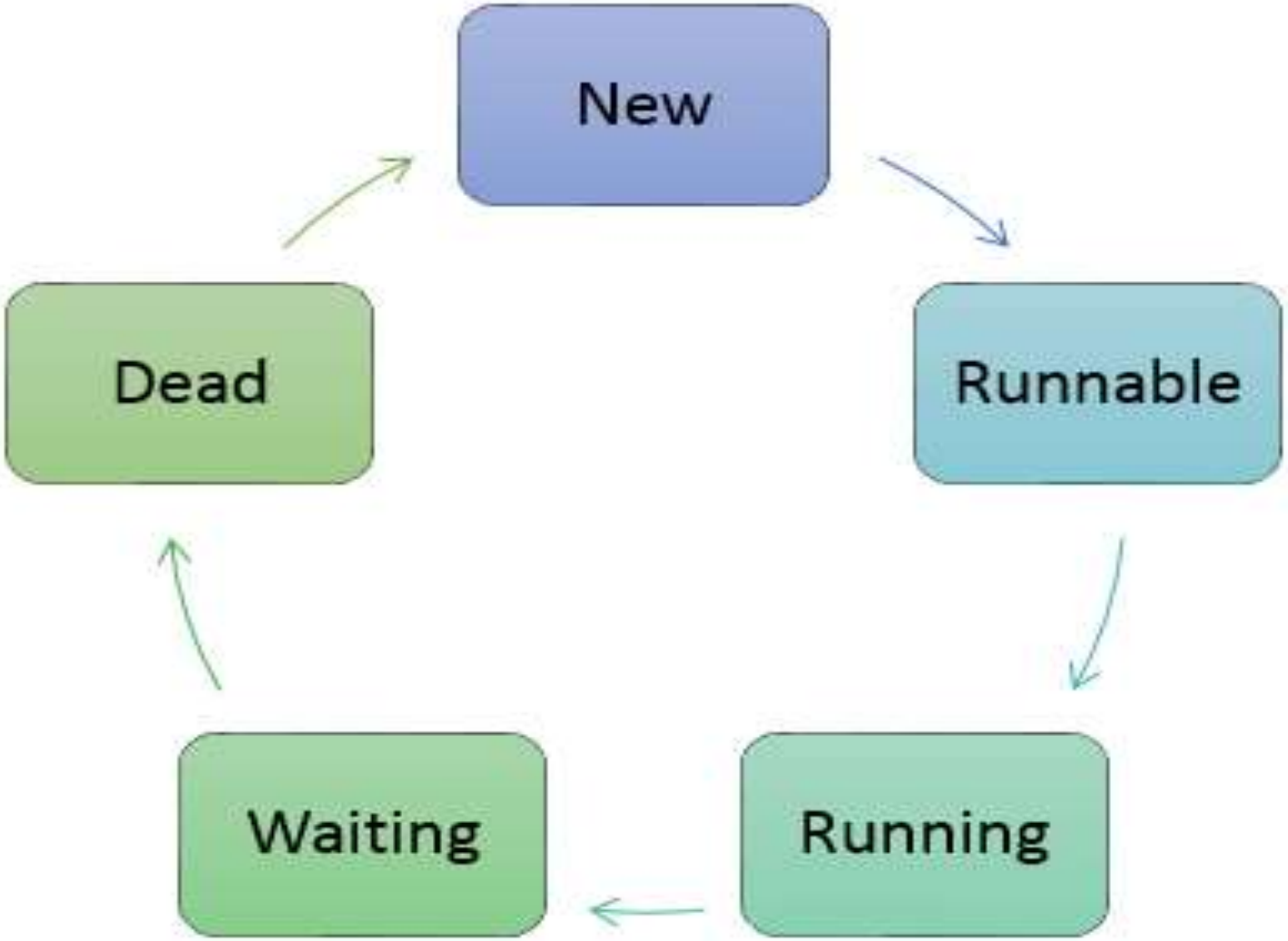
- A finally block appears at the end of the catch blocks and has the following syntax −

```
try {
   // Protected code
} catch (ExceptionType1 e1) {
   // Catch block
} catch (ExceptionType2 e2) {
   // Catch block
} catch (ExceptionType3 e3) {
   // Catch block
}finally {
   // The finally block always executes.
}
```

Multi-threading

- Java is a multi-threaded programming language which means we can develop multi-threaded program using Java.
- **MULTITHREADING** in Java is a process of executing two or more threads simultaneously to maximum utilization of CPU. Multithreaded applications execute two or more threads run concurrently. Hence, it is also known as Concurrency in Java. Each thread runs parallel to each other. Mulitple threads don't allocate separate memory area, hence they save memory.
- Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.
- Threads can be created by using two mechanisms :
  1. Extending the Thread class
  2. Implementing the Runnable Interface

Thread Life Cycle in Java



- There are two ways to create a thread in Java:
  1) By extending Thread class.
  2) By implementing Runnable interface.

Lets Practice…….

Executor Framework

- A framework having a bunch of components that are used for managing worker threads efficiently is referred to as Executor Framework.

- The Executor API reduces the execution of the task from the actual task to be executed through the Executors.

- The executor framework is an implementation of the Producer-Consumer pattern.

- The java.util.concurrent.Executors class provides a set of methods for creating ThreadPools of worker threads.

Types of Executors