

globsyn

Taking People To The Next Level



globalsyn



globsyn 
finishing school

Programming Fundamentals



| Programming fundamentals

- ♣ Topics to be covered in this session
 - OOPS Concepts
 - Inheritance
 - Exception Handling
 - Packages and Interfaces
 - JVM & .jar Files
 - Multithreading

Introduction

- Android is a powerful mobile OS designed for touch screen smartphones
- It runs on Linux platforms
- The program language used here is Java
- In this session we will see the basics of Java concepts

Java Concepts

- | Java is based on Object Oriented Programming Language
- | Hence it involves the basic concepts of OOPS and other advanced concepts like multithreading and .jar files

Java Concepts Contd.

In this session we shall see the following fundamental concepts of OOPS in Java

- ♣ Objects and classes
- ♣ Polymorphism
- ♣ Encapsulation
- ♣ Abstraction
- ♣ Inheritance
- ♣ Exception handling

OOPS Concepts – Objects & Classes

- | Objects are physical entities like desk, chair, table etc.
- | Software objects are used to model these real world objects
- | There are two characteristics for an object
 - ♣ State
 - ♣ Behaviour

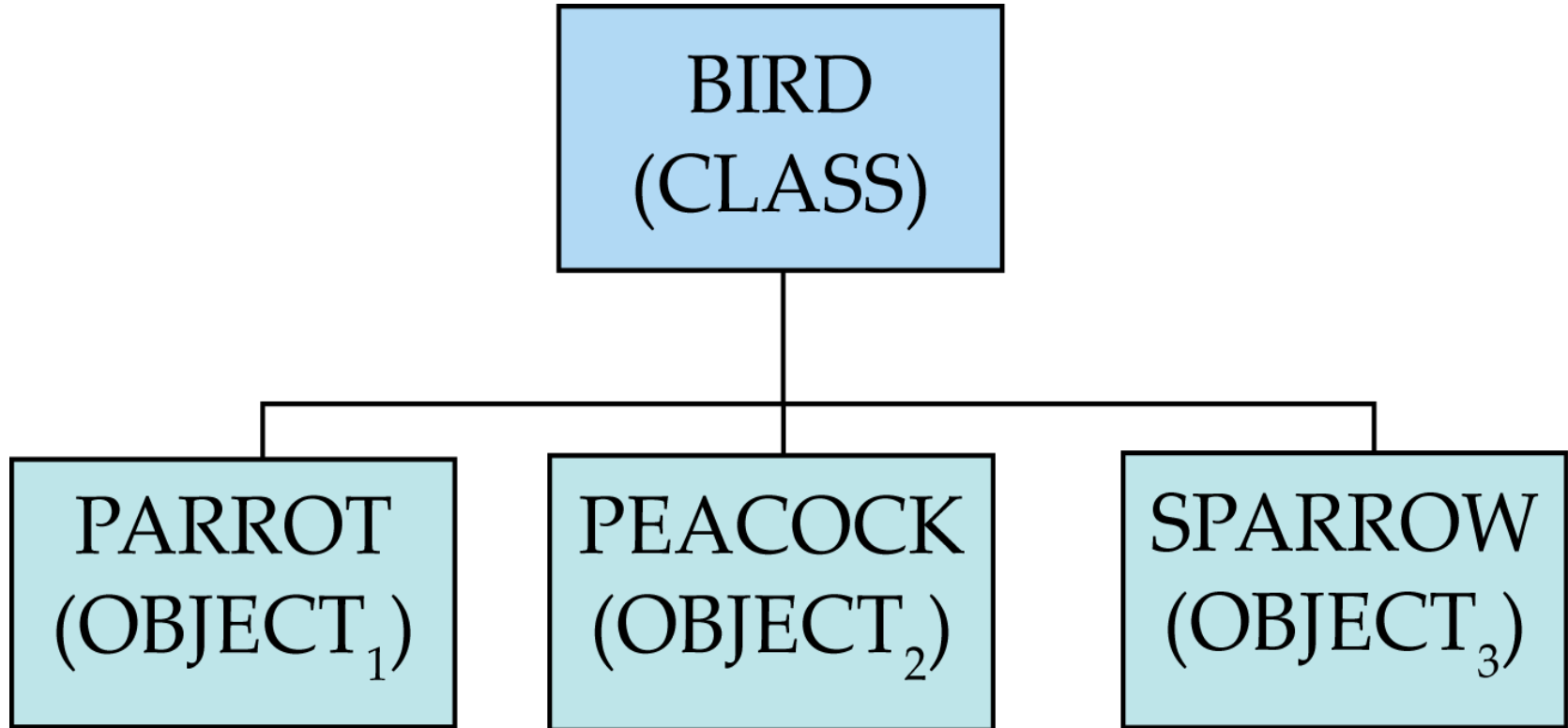
OOPS Concepts – Objects & Classes Contd.

- 1 The state of an object describes for example the colour, breed and texture etc of a dog
- 2 The behaviour of an object refers to for example the barking, fetching, wagging tail etc of a dog
- 3 The states of an object are stored in “fields”
- 4 The behaviour of an object is exhibited by “methods”
- 5 Methods work on an object’s state and facilitate their communication

Classes

- } A class is a blueprint or prototype from which objects are created hence an object is an instance of class
- } A class can contain any of the following variable types
- } **Local variables:** These are defined inside methods, constructors or blocks are called local variables
- } **Instance variables:** Instance variables are variables within a class but outside any method
- } **Class variables:** Class variables are declared within a class and outside of any method, with a static keyword

Objects and Classes



Relationship between objects and classes

Polymorphism

- Polymorphism in simple terms refers to the ability to take different forms
- In OOPS, polymorphism extends the functionality of a super class to certain sub classes which are added at a later period of time to the program
- Hence here an 'extend' keyword is used
- This concept adds adaptability to the existing codes

Abstraction

- 1 The ability to make a class abstract is termed as abstraction
- 2 An abstraction class cannot be instantiated
- 3 Its functionality remains the same even if it does not support instantiation
- 4 In order to declare a class to be an abstract class the keyword abstract is used

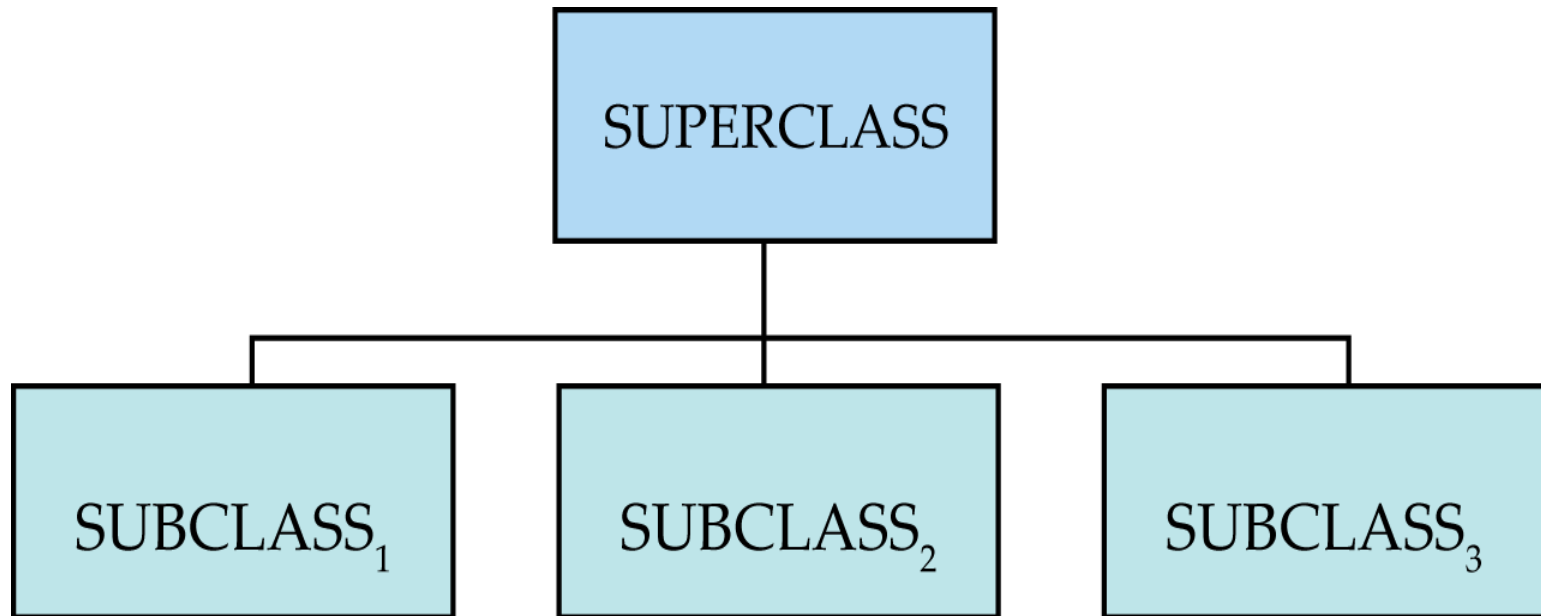
Encapsulation

- Encapsulation is the technique of data hiding
- This is achieved by declaring the class either private or public
- When a class is declared public the fields of this class are said to be accessible to other classes
- When a class is private the fields of this are inaccessible to other classes
- It aids to modify an implemented code without breaking the other codes which use the implemented code

Inheritance

- | mechanism for organizing and structuring the software is known as inheritance
- | When two or more objects have some common attributes they are grouped under a superclass
- | The subclass are those that inherit the attributes of the superclass
- | Each subclass is allowed to have one direct superclass and one superclass can have many subclasses

Inheritance Contd.



Architecture of Inheritance

Exception Handling

- | The execution of Java programs may encounter certain problems these are termed as exceptions
- | It occurs in a Java program due to three main reasons they include the following
 - ♣ User error
 - ♣ Program error
 - ♣ Connection or loss of memory error

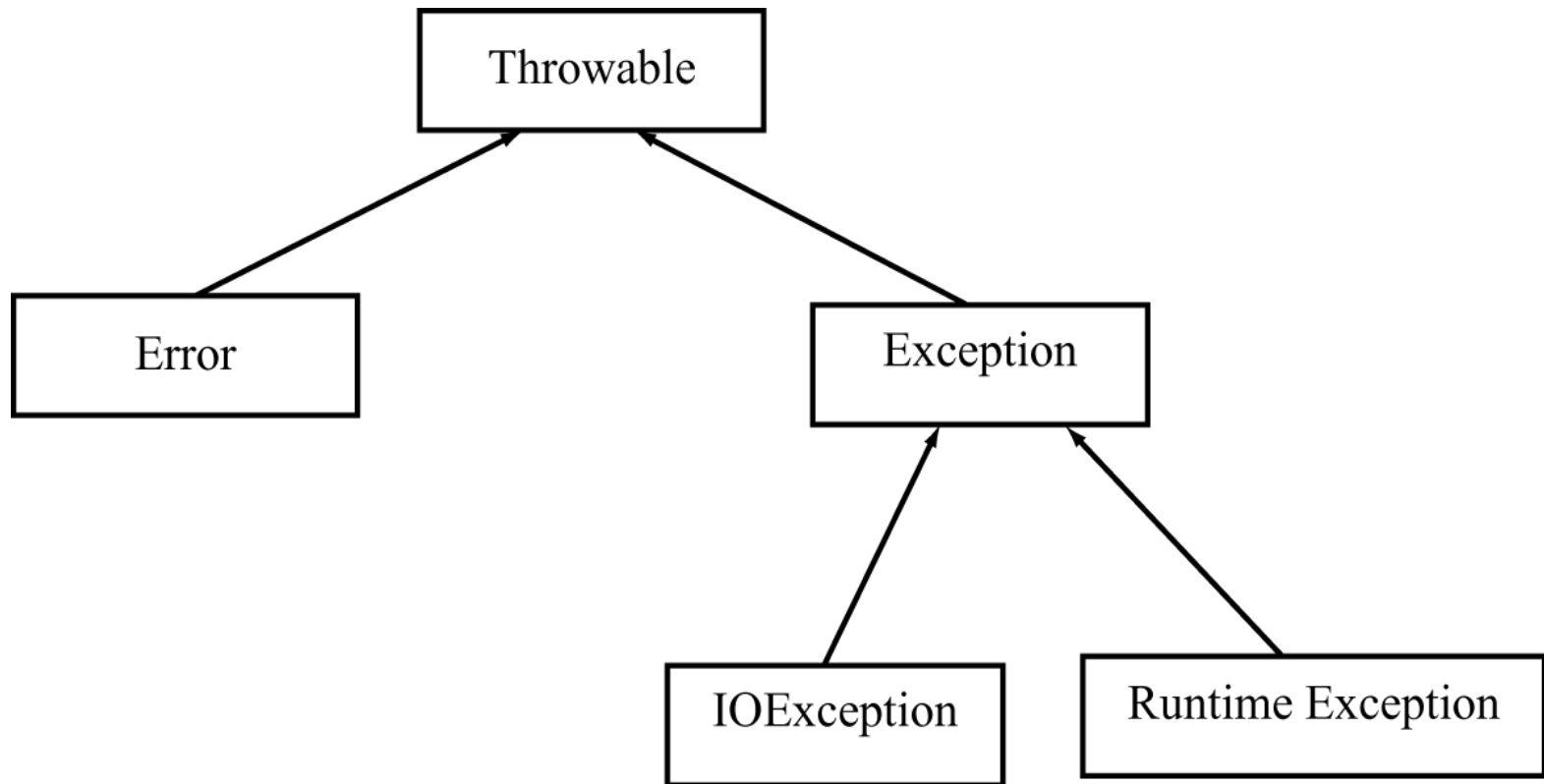
Exception Handling Contd.

- | There are three types of exceptions they include:
- | **Checked exception:** This occurs due to an error committed by an user
- | These exceptions cannot be ignored
- | **Run time exceptions:** This occurs due to an error committed by a programmer
- | They can be neglected during the compilation
- | **Errors:** These are problems that cannot be rectified and is generally ignored

Exception Hierarchy

- | When an exception occurs it is said to be throwable
- | Hence the throwable class is divided into two subclasses
 - ♣ Errors
 - ♣ Exception
- | The exception subclass is further classified into
 - ♣ IO Exception
 - ♣ Runtime Exception

Diagram Representation



It shows the process flow of Exception Hierarchy

Exception handling Contd.

There are two types of keywords used they are:

- ♣ Try and catch
- ♣ Throw

Try and catch: This helps in catching the exceptions that occur within a protected code

Throw: This aids in throwing an occurred exception

Interface

- 1 An interface defines a 'contract' which a class must stick to
- 2 This contracts simply spells out the way how the software will interact
- 3 An interface can only be implemented by classes or can be extended by other interfaces
- 4 Interfaces provide an alternative to multiple inheritance as they are not present in Java

Interface Contd.

- 1 An interface contains behaviors that a class implements
- 2 The interface keyword is used to declare an interface
- 3 When a class implements an interface it signs a contract, agreeing to perform the specific behaviors of the interface
- 4 Classes use the keyword 'implements' to implement an interface

Packages

- 1 Packages refers to the way of organizing files into different directories based on functionality and usability
- 2 The benefits of package reflects the ease of maintenance and improves collaboration
- 3 It helps in managing and using .jar files in more efficient ways
- 4 It helps in preventing collision among the same class names

Packages Contd.

- | The Java platform provides a set of enormous class library (packages) suitable for use in different applications. This library is known as the "Application Programming Interface" (API)
- | API packages are used for general-purpose programming
- | These packages help the programmer to focus on the application at hand

.jar Files

- ┌ .jar files are those that aggregate many files in to one single file
- ┌ jar is an extension of JAVA ARCHIVE
- ┌ Using a .jar file offers the following benefits they include the following:
 - ♣ **Security:** The contents of a .JAR file can be signed digitally
 - ♣ **Decreased download time:** If the apps files along with its resources are bundled in a .jar file downloading the files become easier with a single browser

.jar Files Contd..

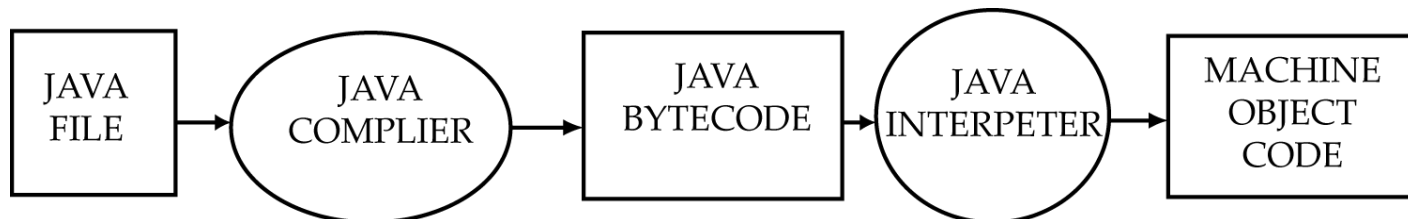
- ♣ **Compression:** Efficiency in storage can be accomplished by compressing the file size
- ♣ **Packaging for extensions:** The software in use can be turned to extensions by using the .jar file
- ♣ **Package Versioning:** The .jar files hold information about the files they contain like the vendor and version information
- ♣ **Portability:** The .jar files have a standard mechanism for facilitating their portability

Java Architecture

Java architecture consists of various components like the following:

- ♣ Java programming language
- ♣ Java class file or .jar file
- ♣ Java virtual machine(JVM)
- ♣ Java application programming interface(API)

The following figure shows the relationship between different components within a java architecture:



Java Architecture Contd..

- 1 The java programs are saved with a .java file extension
- 2 The java compiler converts these .java files to the .class files that contains the bytecodes
- 3 The JVM is nothing but a Java Virtual Machine that converts the bytecode of the java language into machine understandable form

Java Architecture Contd..

- For each new operating system environment, JVM needs to be implemented uniquely
- Lastly, the Java Application Programming Interface (API) serves as a communicating medium for the JVM to interact with the hardware present

Java Virtual Machine (JVM)

- | JVM forms the base for the JAVA platform and is convenient to use on any hardware
- | The components of a JVM are:
 - ♣ Class loader
 - ♣ Execution engine
 - ♣ Just In Time (JIT) compiler
- | **Class loader:** It loads the java class files dynamically when required by a program from the memory. There are two types of class loaders they include
 - | Primordial class loader
 - | Class loader objects

Java Virtual Machine Contd..

- 1 **Primordial class loaders:** It loads the JAVA API classes required by the running java program
- 2 **Class loader objects:** Loads the classes of the application program
- 3 **Execution engine:** This component runs the bytecode in a sequential manner(line by line). This is the crucial component which aids the JVM to convert the bytecode to the machine understandable language

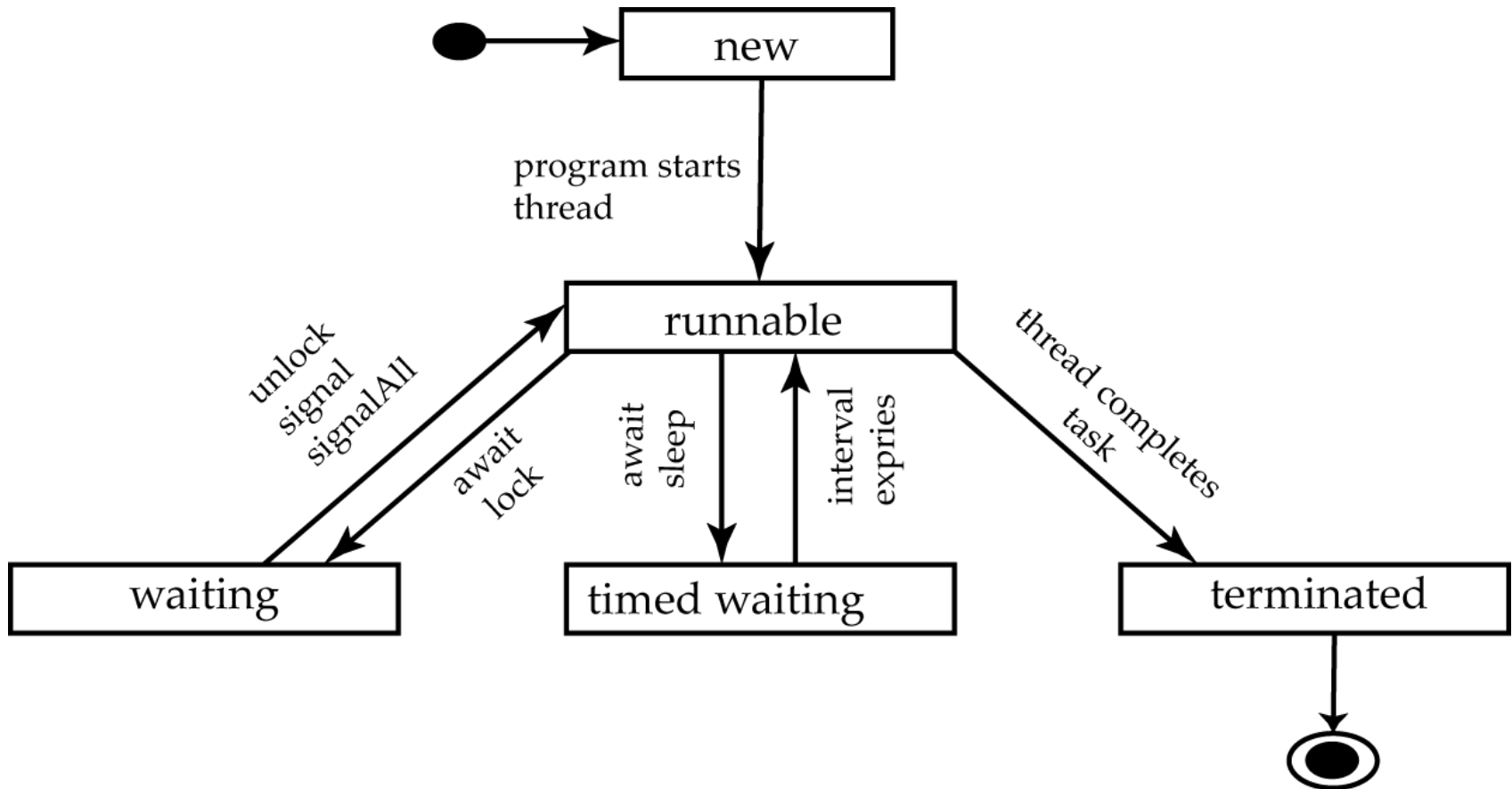
Java Virtual Machine Contd..

JIT compiler: This is a compiler that is used to run the machine code. Unlike an execution engine JIT does not run in a sequential manner and hence is faster than its counterpart

Multithreading

- 1 Multithreading is a concept wherein a program contains two or more parts that runs concurrently
- 2 Each part of such a program is called a thread, and each thread contains a separate execution path
- 3 A thread or a couple of threads are contained within a memory space allocated by the operating system
- 4 The advantage of multithreading is that it enables to efficiently develop programs that maximizes the use of the CPU by reducing the idle time
- 5 A thread has various stages of life and it is described with the aid of a flowchart as shown below

Life Cycle of a Thread



Life Cycle of a Thread Contd..


- 1 **New:** This is the stage when a new thread is said to be born and the thread remains in this state until the program starts it
- 2 **Runnable:** Once a program starts a thread it becomes runnable where it is said to be executing its task
- 3 **Waiting:** When one of the thread is currently executing its task an interrupting thread goes to a waiting state for the ongoing execution to continue. When this completes the interrupting thread resumes back from its waiting state

Life Cycle of a Thread Contd..

- 1 **Timed waiting:** This is a state wherein a runnable thread waits for a stipulated interval of time. When this stipulated time expires the thread resumes back its runnable state or when the event it is waiting for occurs
- 2 **Terminated:** In the event of completion of the assigned task a runnable thread completely terminates itself
- 3 **Thread Priorities:** The operating system has a thread schedule based on which each and every thread is executed. The thread scheduling is based on the priorities assigned to the threads

Life Cycle of a Thread Contd..

- 1 The priorities are ranged between Min_Priority (1) and Max_Priority (10). Every thread is given a Norm_Priority (5) by default
- 2 Higher priority threads are significant when allocating processor time when compared to lower priority
- 3 However, the priorities do not guarantee the order of the execution of the threads and very much platform dependent



Taking People To The Next Level ...