# Multithreading

Execution of more than one thread at a time is called multi-threading.

* A thread is a piece of code that executes independently
* Every program contains at-least one thread that is the main thread.
* Main thread default priority is a normal priority, that is number 5.

There are 2 ways to create a new thread.

1. By extending java.lang.Thread class
2. By implementing java.lang.Runnable interface

Whenever the thread class constructor is called then a new thread will be born. Whenever destroy method is called the thread will die.

The sleep method takes milliseconds as a parameter.

1000 milliseconds = 1 second.

**Diagram

Description automatically generated**

**Diagram, timeline

Description automatically generated**

* Run method is given as null body method in a thread class to write child thread task code.
* The start() method implicitly calls the run() method.

**Java.lang.Thread**

**Fields(Variables):**

**Public static final int MIN\_PRIORITY; (1)**

**Public static final int NORM\_PRIORITY; (5)**

**Public static final int MAX\_PRIORITY; (10)**

**Constructors:**

**Public Thread();**

**Public Thread(Runnable);**

**Methods:**

**Public static native Thread currentThread();**

**Public static native void sleep(long) throws interruptedException;**

**Public synchronized void start();**

**Public void run();**

**Public void destroy();**

**Public final void suspend();**

**Public final void resume();**

**Public final void setPriority(int);**

**Public final void getPriority ();**

**Public final synchronized void setName(String);**

**Public final String getName();**

Refer code snippet: MainThreadExample

|  |
| --- |
| //Program to demonstrate main thread  **package** multithreading;  **import** java.lang.Thread;  **public** **class** MainThreadExample **extends** Thread  {  **public** **static** **void** main(String[] args)  {  Thread t = Thread.*currentThread*();  System.***out***.println(t.getName());  System.***out***.println(t.getPriority());    t.setName("Main Thread");  t.setPriority(10);    System.***out***.println(t.getName());  System.***out***.println(t.getPriority());  System.***out***.println(t.getState());  }  }  //OUTPUT  main  5  Main Thread  10  RUNNABLE |

Refer code snippet: SleepMethodExample

|  |
| --- |
| //Program to demonstrate sleep() method  **package** multithreading;  **import** java.lang.Thread;  **public** **class** SleepMethodExample **extends** Thread  {  **public** **static** **void** main(String[] args) **throws** InterruptedException  {  **for** (**int** i =1; i<=10;i++)  {  System.***out***.println(i);  Thread.*sleep*(1000l);  }  }  }  //OUTPUT  1  2  3  4  5  6  7  8  9  10 |

Graphical user interface, diagram, application

Description automatically generated

Steps to Develop Multithreaded application by extending java.lang.Thread class

* Create a class that extends java.lang.Thread class
* Override run method.
* Write child thread class code in run() method.
* Write main() method.
* Create an object of current class.
* Call start() method.
* Write main thread task code in main() method.

Refer code snippet: MuttiThreadApplicationExample1

|  |
| --- |
| **//Program to demonstrate multithreaded application using java.lang.Thread class**  **package** multithreading;  **import** java.lang.Thread;  **public** **class** MuttiThreadApplicationExample1 **extends** Thread  {  **public** **void** run()  {  **for** (**int** i =1;i<=10;i++)  {  System.***out***.println("Child Thread - "+i);  }  }  **public** **static** **void** main(String[] args)  {  MuttiThreadApplicationExample1 obj = **new** MuttiThreadApplicationExample1();  obj.start();    /\* Main Thread Code\*/  **for** (**int** i =1;i<=10;i++)  {  System.***out***.println("Main Thread - "+i);  }  }  }  //Output  Main Thread - 1  Main Thread - 2  Child Thread - 1  Main Thread - 3  Child Thread - 2  Main Thread - 4  Child Thread - 3  Main Thread - 5  Child Thread - 4  Main Thread - 6  Child Thread - 5  Main Thread - 7  Child Thread - 6  Main Thread - 8  Child Thread - 7  Main Thread - 9  Child Thread - 8  Main Thread - 10  Child Thread - 9  Child Thread - 10 |

Refer code snippet: MultiThreadedApplicationwithMoreThan2ThreadsExample1

|  |
| --- |
| **//Program to demonstrate more than 2 threads.**  **package** multithreading;  **import** java.lang.Thread;  **public** **class** MultiThreadedApplicationwithMoreThan2ThreadsExample1 **extends** Thread  {  **public** **void** run()  {  **for** (**int** i =1;i<=10;i++)  {  System.***out***.println(getName()+" - "+i);  }  }  **public** **static** **void** main(String[] args)  {  MultiThreadedApplicationwithMoreThan2ThreadsExample1 obj1 = **new** MultiThreadedApplicationwithMoreThan2ThreadsExample1();  MultiThreadedApplicationwithMoreThan2ThreadsExample1 obj2 = **new** MultiThreadedApplicationwithMoreThan2ThreadsExample1();    // Call Child Thread 1  obj1.setName("Child Thread1");  obj1.start();    //Call Child Thread2  obj2.setName("Child Thread2");  obj2.start();    //Main thread  **for** (**int** i =1;i<=10;i++)  {  System.***out***.println("Main Thread - "+i);  }  }  }  //output  Main Thread - 1  Main Thread - 2  Main Thread - 3  Child Thread1 - 1  Child Thread1 - 2  Child Thread2 - 1  Child Thread1 - 3  Child Thread1 - 4  Main Thread - 4  Main Thread - 5  Main Thread - 6  Child Thread1 - 5  Child Thread1 - 6  Child Thread1 - 7  Child Thread2 - 2  Child Thread2 - 3  Child Thread1 - 8  Main Thread - 7  Main Thread - 8  Main Thread - 9  Main Thread - 10  Child Thread1 - 9  Child Thread2 - 4  Child Thread2 - 5  Child Thread2 - 6  Child Thread2 - 7  Child Thread2 - 8  Child Thread1 - 10  Child Thread2 - 9  Child Thread2 - 10 |

Refer code snippet: MutiThreadedApplicationwithMoreThan2ThreadswithdifferentSleeptime

|  |
| --- |
| **//Program to demonstrate more than 2 threads with different sleep times**  **package** multithreading;  **import** java.lang.Thread;  **public** **class** MutiThreadedApplicationwithMoreThan2ThreadswithdifferentSleeptime **extends** Thread  {  **public** **void** run()  {  **for** (**int** i=1;i<=10;i++)  {  **try**  {  **if** (getName().equalsIgnoreCase("Child Thread1"))  {  System.***out***.println("Child Thread1 - "+i);  Thread.*sleep*(1000);  }  **else** **if** (getName().equalsIgnoreCase("Child Thread2"))  {  System.***out***.println("Child Thread2 - "+i);  Thread.*sleep*(2000);  }  }  **catch** (InterruptedException e)  {  e.getMessage();  }  }  }  **public** **static** **void** main(String[] args)  {  MutiThreadedApplicationwithMoreThan2ThreadswithdifferentSleeptime obj1 = **new** MutiThreadedApplicationwithMoreThan2ThreadswithdifferentSleeptime();  MutiThreadedApplicationwithMoreThan2ThreadswithdifferentSleeptime obj2 = **new** MutiThreadedApplicationwithMoreThan2ThreadswithdifferentSleeptime();    //Call ChildThread1  obj1.setName("Child Thread1");  obj1.start();    //Call ChildThread2  obj2.setName("Child Thread2");  obj2.start();    //Main thread  **try**  {  **for** (**int** i=1;i<=10;i++)  {  System.***out***.println("Main Thread - "+i);  Thread.*sleep*(3000);  }  }  **catch** (InterruptedException ex)  {  ex.getMessage();  }  }  }  //OUTPUT  Main Thread - 1  Child Thread1 - 1  Child Thread2 - 1  Child Thread1 - 2  Child Thread1 - 3  Child Thread2 - 2  Child Thread1 - 4  Main Thread - 2  Child Thread2 - 3  Child Thread1 - 5  Child Thread1 - 6  Child Thread1 - 7  Main Thread - 3  Child Thread2 - 4  Child Thread1 - 8  Child Thread2 - 5  Child Thread1 - 9  Main Thread - 4  Child Thread1 - 10  Child Thread2 - 6  Main Thread - 5  Child Thread2 - 7  Child Thread2 - 8  Main Thread - 6  Child Thread2 - 9  Main Thread - 7  Child Thread2 - 10  Main Thread - 8  Main Thread - 9  Main Thread - 10 |

Refer code snippet: SuspendAndResumeMethodsExample

|  |
| --- |
| **//Program to demonstrate suspend and resume methods**  **package** multithreading;  **import** java.lang.Thread;  **public** **class** SuspendAndResumeMethodsExample **extends** Thread  {  @SuppressWarnings("deprecation")  **public** **void** run()  {  **try**  {  **for** (**int** i=1;i<=10;i++)  {  System.***out***.println("Child Thread - "+i);  Thread.*sleep*(1000);  **if** (i == 5)  {  ~~suspend~~();  }  **if** (i==9)  {  ~~resume~~();  }  }  }  **catch**(InterruptedException ex)  {  ex.getMessage();  }  }  **public** **static** **void** main(String[] args) **throws** InterruptedException  {  SuspendAndResumeMethodsExample obj = **new** SuspendAndResumeMethodsExample();  obj.start();    //Main thread  **for** (**int** i=1;i<=10;i++)  {    System.***out***.println("Main Thread - "+i);  Thread.*sleep*(1000);    }  }  }  //OUTPUT  Main Thread - 1  Child Thread - 1  Child Thread - 2  Main Thread - 2  Main Thread - 3  Child Thread - 3  Main Thread - 4  Child Thread - 4  Child Thread - 5  Main Thread - 5  Main Thread - 6  Main Thread - 7  Main Thread - 8  Main Thread - 9  Main Thread - 10 |

Steps to create a multithreaded application by implementing java.lang.Runnable interface

* Create a class that implements java.lang.Runnable interface
* Override run() method
* Write child thread task code in run() method
* Write main() method
* Create an object of the current class and assign it to a Runnable reference
* Create an object of thread class by passing Runnable reference in Thread class constructor.
* Call start() method.
* Write main thread task code in main() method.

Refer code snippet: MultiThreadingApplicationWithRunnableInterface

|  |
| --- |
| **//Program to demonstrate multithreaded application by implementing java.lang.Runnable interface**  **package** multithreading;  **public** **class** MultiThreadingApplicationWithRunnableInterface **implements** Runnable  {  **public** **void** run()  {  **for** (**int** i =1;i<=10;i++)  {  System.***out***.println("Child Thread - "+i);  }  }  **public** **static** **void** main(String[] args)  {  Runnable r = **new** MultiThreadingApplicationWithRunnableInterface();  Thread t = **new** Thread(r);  t.start();    //Main Thread  **for** (**int** i =1;i<=10;i++)  {  System.***out***.println("Main Thread - "+i);  }  }  }  //OUTPUT  Main Thread - 1  Main Thread - 2  Main Thread - 3  Child Thread - 1  Main Thread - 4  Main Thread - 5  Main Thread - 6  Main Thread - 7  Main Thread - 8  Child Thread - 2  Child Thread - 3  Child Thread - 4  Main Thread - 9  Main Thread - 10  Child Thread - 5  Child Thread - 6  Child Thread - 7  Child Thread - 8  Child Thread - 9  Child Thread - 10 |