Advanced Programming Language

Assignment 6

Chinmay Dorge

LCS2020022

#1) Java Thread life Cycle

Create Java program demonstrating thread states. Create three threads and name them as sachin, virat and sehwag. Using java.lang.Thread.class demonstrate life cycle of the thread, i.e.

1. New 2) Active 3) Runnable 4) Running 5) Blocked/Waiting.

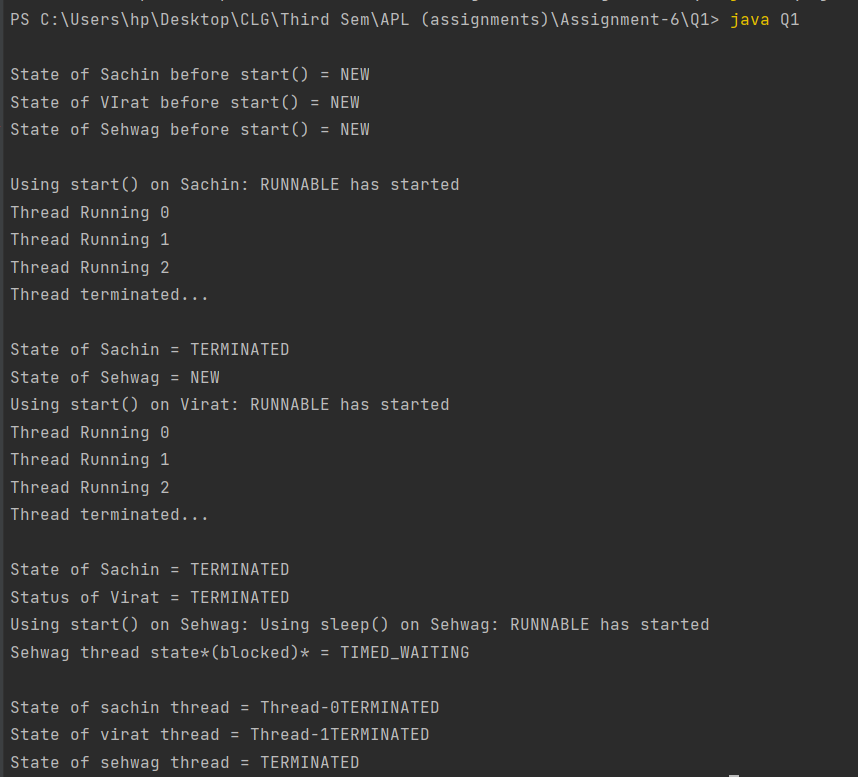
Code: MyThread.java

public class MyThread extends Thread{  
@Override  
public void run(){  
 System.out.println(Thread.currentThread().getState()+" has started");  
 for(int i=0; i<3;i++ )  
 {  
 System.out.println( "Thread Running "+ i);  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 System.out.println("Thread terminated...\n");  
 }  
}

Q1.java

public class Q1{  
 public static void main (String[] args) throws InterruptedException {  
 MyThread sachin = new MyThread();  
 MyThread virat = new MyThread();  
 var sehwag = new Thread(new Runnable() {  
 @Override  
 public void run() {  
 System.out.println(Thread.currentThread().getState()+" has started");  
 try {  
 Thread.sleep(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
  
 System.out.println("\nState of Sachin before start() = "+ sachin.getState());  
 System.out.println("State of VIrat before start() = "+ virat.getState());  
 System.out.println("State of Sehwag before start() = "+ sehwag.getState()+"\n");  
  
 System.out.print("Using start() on Sachin: ");  
 sachin.start();  
 sachin.join();  
  
 virat.sleep(3000);  
  
 System.out.println("State of Sachin = "+ sachin.getState()+"\nState of Sehwag = "+ sehwag.getState());  
 System.out.print("Using start() on Virat: ");  
 virat.start();  
 virat.join();  
  
 System.out.println("State of Sachin = "+ sachin.getState()+"\nStatus of Virat = "+ virat.getState());  
  
 System.out.print("Using start() on Sehwag: ");  
 sehwag.start();  
 System.out.print("Using sleep() on Sehwag: ");  
 Thread.sleep(10);  
 System.out.println("Sehwag thread state\*(blocked)\* = "+ sehwag.getState());  
 sehwag.join();  
  
 System.out.println("\nState of sachin thread = "+sachin.getName()+sachin.getState());  
 System.out.println("State of virat thread = "+virat.getName()+virat.getState());  
 System.out.println("State of sehwag thread = "+sehwag.getState());  
 }  
}

Output:



#2) Java Thread Methods

Create Java program that implements 3 threads using Thread class. Name them as Sania,

Maria, Serena. Give them different priority and demonstrate the use of following methods:

1) run()

2) start()

3) sleep() -Sania sleeps for 100ms,Maria for 200ms,Serena for 300ms

4) join()

5) getpriority()

6) yield()

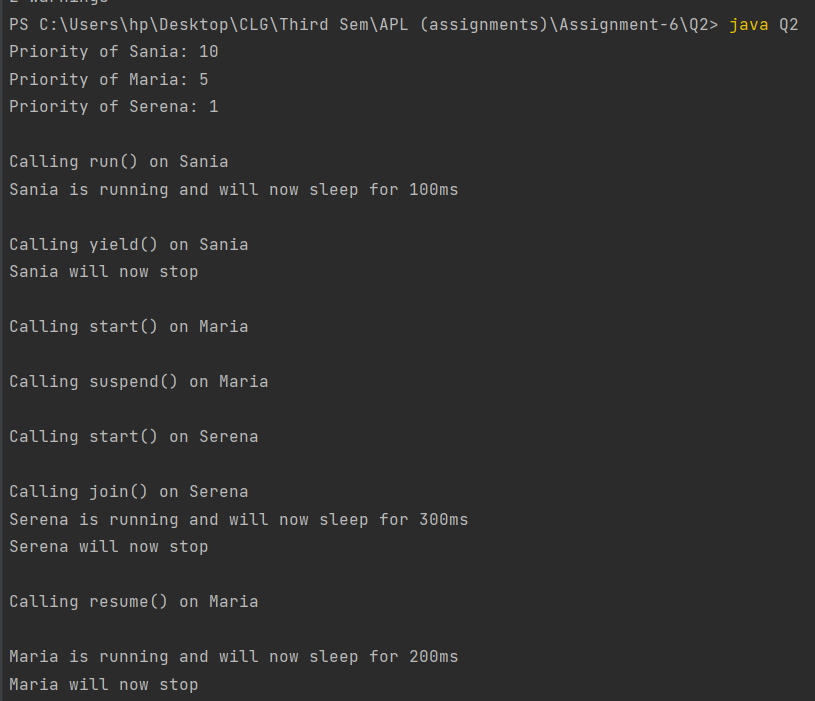
7) suspend

8) Resume

Code: Q2.java

public class Q2 {  
 public static void main(String[] args) {  
 Thread sania = new Thread() {  
 public void run() {  
 System.out.println("Sania is running and will now sleep for 100ms");  
 try {  
 sleep(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.out.println();  
 System.out.println("Calling yield() on Sania");  
 Thread.yield();  
 System.out.println("Sania will now stop");  
 }  
 };  
 Thread maria = new Thread() {  
 public void run() {  
 System.out.println("Maria is running and will now sleep for 200ms");  
 try {  
 sleep(200);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.out.println("Maria will now stop");  
 }  
 };  
 Thread serena = new Thread() {  
 public void run() {  
 System.out.println("Serena is running and will now sleep for 300ms");  
 try {  
 sleep(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.out.println("Serena will now stop");  
 }  
 };  
  
 sania.setPriority(Thread.MAX\_PRIORITY);  
 maria.setPriority(Thread.NORM\_PRIORITY);  
 serena.setPriority(Thread.MIN\_PRIORITY);  
  
 System.out.println("Priority of Sania: " + sania.getPriority());  
 System.out.println("Priority of Maria: " + maria.getPriority());  
 System.out.println("Priority of Serena: " + serena.getPriority());  
 System.out.println();  
  
 System.out.println("Calling run() on Sania");  
 sania.run();  
 System.out.println();  
  
 System.out.println("Calling start() on Maria");  
 maria.start();  
 System.out.println();  
  
 System.out.println("Calling suspend() on Maria");  
 maria.suspend();  
 System.out.println();  
  
 System.out.println("Calling start() on Serena");  
 serena.start();  
 System.out.println();  
  
 try {  
 System.out.println("Calling join() on Serena");  
 serena.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.out.println();  
  
 System.out.println("Calling resume() on Maria");  
 maria.resume();  
 System.out.println();  
 }  
}

Output:



#3) Start vs run

Implement the above program using runnable interface. Also, see what happens when you call run method directly without calling start. And set equal priority to all threads. Write a program that demonstrates this and print your explanation in the terminal for what is happening when you

call run() directly.

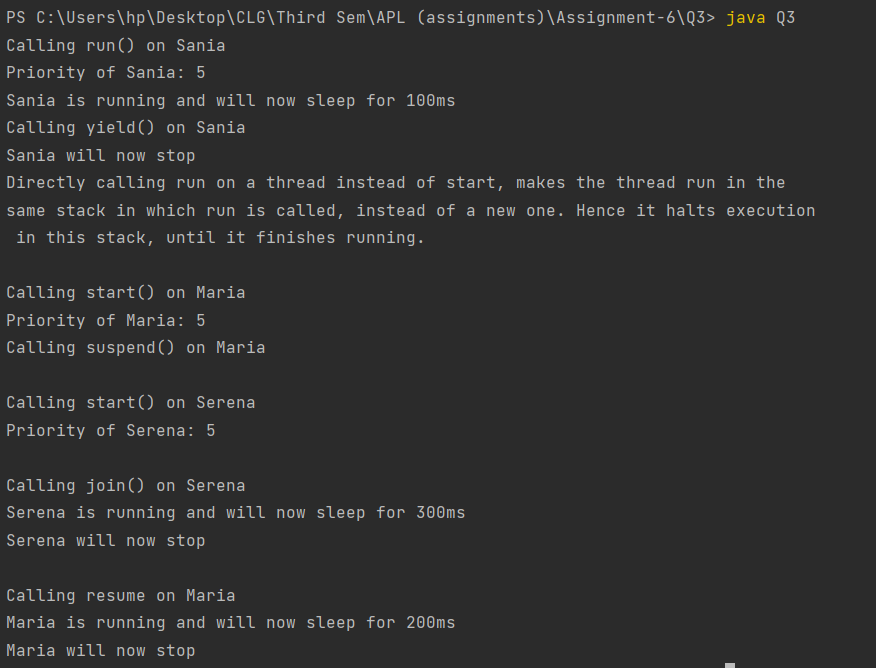
Code: Q3.java

class SaniaRunnable implements Runnable {  
 public void run() {  
 System.out.println("Sania is running and will now sleep for 100ms");  
 try {  
 Thread.sleep(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.out.println("Calling yield() on Sania");  
 Thread.yield();  
 System.out.println("Sania will now stop");  
 }  
}  
  
class MariaRunnable implements Runnable {  
 public void run() {  
 System.out.println("Maria is running and will now sleep for 200ms");  
 try {  
 Thread.sleep(200);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.out.println("Maria will now stop");  
 }  
}  
  
class SerenaRunnable implements Runnable {  
 public void run() {  
 System.out.println("Serena is running and will now sleep for 300ms");  
 try {  
 Thread.sleep(300);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.out.println("Serena will now stop");  
 }  
}  
  
public class Q3 {  
 public static void main(String[] args) {  
 Thread sania = new Thread(new SaniaRunnable());  
 Thread maria = new Thread(new MariaRunnable());  
 Thread serena = new Thread(new SerenaRunnable());  
  
 sania.setPriority(5);  
 maria.setPriority(5);  
 serena.setPriority(5);  
  
 System.out.println("Calling run() on Sania");  
 System.out.println("Priority of Sania: " + sania.getPriority());  
 sania.run();

System.out.println("Directly calling run on a thread instead of start, makes the thread run in the same stack in which run is called, instead of a new one. Hence it halts execution in this stack, until it finishes running.");

System.out.println();  
  
 System.out.println("Calling start() on Maria");  
 maria.start();  
 System.out.println("Priority of Maria: " + maria.getPriority());  
 System.out.println("Calling suspend() on Maria");  
 maria.suspend();  
 System.out.println();  
  
 System.out.println("Calling start() on Serena");  
 System.out.println("Priority of Serena: " + serena.getPriority());  
 serena.start();  
 System.out.println();  
  
 try {  
 System.out.println("Calling join() on Serena");  
 serena.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.out.println();  
  
 System.out.println("Calling resume on Maria");  
 maria.resume();  
 }  
}

Output:



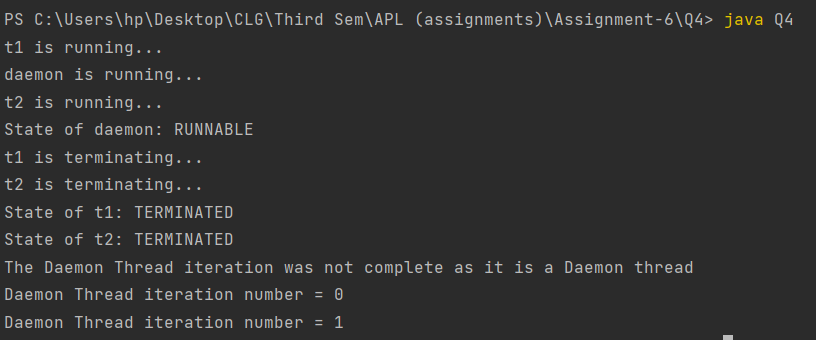
#4) Daemon Thread

“Daemon Thread are threads who sole purpose is to serve other threads. When it is no longer serving anyone it dies.” Create a java program that implements 3 threads. Make one of them as daemon thread and demonstrate the truth of the above statement.

Code: Q4.java

class MyRunnable implements Runnable {  
 public void run() {  
 System.out.println(Thread.currentThread().getName() + " is running...");  
 for(int i = 0; i < 1000000000; i++) {}  
 System.out.println(Thread.currentThread().getName() + " is terminating...");  
 }  
}  
  
class DaemonRunnable implements Runnable {  
 public void run() {  
 System.out.println(Thread.currentThread().getName() + " is running...");  
 for(int i = 0; i < 1000000000; i++) {  
 System.out.println("Daemon Thread iteration number = " + i);  
 }  
 }  
}  
  
public class Q4 {  
 public static void main(String[] args) {  
 MyRunnable myRunnable = new MyRunnable();  
 DaemonRunnable daemonRunnable = new DaemonRunnable();  
   
 Thread daemon = new Thread(daemonRunnable, "daemon");  
 daemon.setDaemon(true);  
   
 Thread t1 = new Thread(myRunnable, "t1");  
 Thread t2 = new Thread(myRunnable, "t2");  
   
 t1.start();  
 t2.start();  
 daemon.start();  
   
 System.out.println("State of daemon: " + daemon.getState());  
 try {  
 t1.join();  
 t2.join();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 System.out.println("State of t1: " + t1.getState());  
 System.out.println("State of t2: " + t2.getState());  
  
 System.out.println("The Daemon Thread iteration was not complete as it is a Daemon thread ");  
 }  
}

Output:



#5) Thread Poll/ThreadGroup

Create a Java program that implements three thread classes in different packages. In the main methods create instances of it and put it in a ThreadGroup. Demonstrate any 7 methods available in Java Thread Group Class.

Code: BgSYNCThread.java

package dataprocess;  
  
public class BgSYNCThread extends Thread {  
 public BgSYNCThread(ThreadGroup tg, String name) {  
 super(tg, name);  
 }  
 public void run() {  
 System.out.println("Syncing all the data of the app with the cloud");  
 for(int i = 0; i < 1000000000; i++);  
 }   
}

NotificationsThread.java

package Notifications;  
  
public class NotificationsThread extends Thread {  
 public NotificationsThread(ThreadGroup tg, String name) {  
 super(tg, name);  
 }  
 public void run() {  
 System.out.println("Checking for and receiving notifications of the app");  
 for(int i = 0; i < 1000000000; i++);  
 }   
}

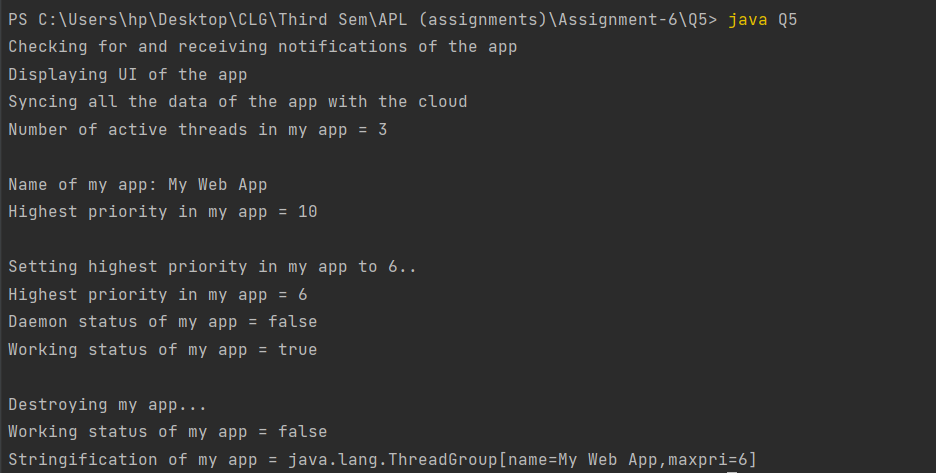
UIThread.java

package Userinterface;  
  
public class UIThread extends Thread {  
 public UIThread(ThreadGroup tg, String name) {  
 super(tg, name);  
 }  
 public void run() {  
 System.out.println("Displaying UI of the app");  
 for(int i = 0; i < 1000000000; i++);  
 }  
}

Q5.java

import Userinterface.UIThread;  
import Notifications.NotificationsThread;  
import dataprocess.BgSYNCThread;  
  
public class Q5 {  
 public static void main(String[] args) {  
 ThreadGroup myApp = new ThreadGroup("My Web App");  
 Thread ui = new UIThread(myApp, "User Interface");  
 Thread notifications = new NotificationsThread(myApp, "Notifications");  
 Thread backgroundSync = new BgSYNCThread(myApp, "Background Sync");  
  
 ui.start();  
 notifications.start();  
 backgroundSync.start();  
  
 Thread[] appThreads = new Thread[myApp.activeCount()];   
 System.out.println("Number of active threads in my app = " + myApp.activeCount());  
 myApp.enumerate(appThreads);  
  
 System.out.println();  
 System.out.println("Name of my app: " + myApp.getName());  
 System.out.println("Highest priority in my app = " + myApp.getMaxPriority());  
 System.out.println();  
  
 myApp.setMaxPriority(6);  
 System.out.println("Setting highest priority in my app to 6..");  
 System.out.println("Highest priority in my app = " + myApp.getMaxPriority());  
 System.out.println("Daemon status of my app = " + myApp.isDaemon());  
 System.out.println("Working status of my app = " + !myApp.isDestroyed());  
 System.out.println();  
 myApp.destroy();  
 System.out.println("Destroying my app...");  
 System.out.println("Working status of my app = " + !myApp.isDestroyed());  
 System.out.println("Stringification of my app = " + myApp.toString());  
 }  
}

Output:



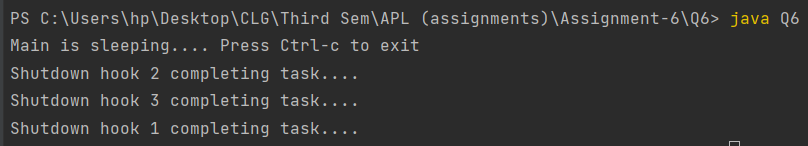
#6) Java Shutdown Hook

“The shutdown hook can be used to perform cleanup resource or save the state when JVM shuts down normally or abruptly.” Write a java program that demonstrates how java shutdown hook works by create 3 anonymous thread classes. Create an instance of runnable and add the threads classes created earlier into the shutdown hook. Demonstrate that the above statement is true.

Code: Q6.java

class Thread1 extends Thread {  
 public void run() {  
 System.out.println("Shutdown hook 1 completing task....");  
 }  
}  
class Thread2 extends Thread {  
 public void run() {  
 System.out.println("Shutdown hook 2 completing task....");  
 }  
}  
class Thread3 extends Thread {  
 public void run() {  
 System.out.println("Shutdown hook 3 completing task....");  
 }  
}  
public class Q6 {  
 public static void main(String[] args) {  
 Runtime r = Runtime.getRuntime();  
 r.addShutdownHook(new Thread1());  
 r.addShutdownHook(new Thread2());  
 r.addShutdownHook(new Thread3());  
 System.out.println("Main is sleeping.... Press Ctrl-c to exit");  
 try  
 {  
 Thread.sleep(2000);  
 }  
 catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

Output:



#7) Garbage Collection and Runtime class

Write a java program that demonstrates the use of java runtime class to following operations

1) Open notepad

2) Create a new file called mythread.java

Also, implement demonstrate garbage collection in java. (Hint: think of differencing and finalize() method);

Code: Q7.java

import java.io.IOException;  
  
public class Q7 {  
 public void finalize() {  
 System.out.println("garbage collection done!");  
 }  
 public static void main(String[] args) throws IOException {  
 Q7 obj = new Q7();  
 obj = null;  
 System.gc();  
 Runtime.getRuntime().exec("notepad mythread.java");  
 }  
}

Note: The code has been sent with the zip file and is also available on GitHub. Repo link

<https://github.com/Chinmay-Dorge/Advanced-Programming-Assignments>