CS 342 : Software Design

March 12th – Threads

Announcements

- Project2 –due Thursday
- HW4 due today
- Project1 / HW4 /Makeup quiz grades back after spring break

Threads

- Process

- Threads

Threads

- **Process: often thought of as an application or program.**
  - Each process has its own private memory space and run-time resources

- **Threads: exist within a process.**
  - Threads share a processes resources including memory and open files
  - Every application has at least one thread
  - Each thread has own runtime stack and registers
  - A thread shares statically and dynamically allocated objects with other threads in the same process
  - A thread is an independent path of execution within a program
  - Can enable two or more tasks to execute concurrently in a single process

https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html

Threads

- In Java, each thread is associated with an instance of the Thread class.
- Each instance is a seperate path of execution in a program.
- The JVM allows any application to have multiple threads of execution running concurrently (at the same time).
- There are two basic methods for creating threads in your program:
  - Extend the Java class Thread
  - Implement the runnable interface

https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html

Threads: Extend the Thread class

- Create a class that extends the Thread class
  - `class MyThread extends Thread{}`
- Implement the abstract method `run()`
  - `public void run(){}`
- The method run() is what will execute when you start the thread.
  - It is said to be in the “running state”
- In your code you can create an instance of your class:
  - `MyThread t1 = new MyThread();`
- Then start the run() method:
  - `t1.start() //this will call the run() method in that instance`
- When the run() method returns, your thread will be terminated
  - It is said to be in the “terminated state”
  - If there are no references to it, the thread can be garbage collected

Benefits of extending the Thread class

- Extending the Thread class gives you access to all of the Thread class methods but it keeps your class from inheriting from any other class.
- If you need to extend some other class, you should implement runnable instead.

Threads: implement runnable interface

- Create a class that implements the Runnable interface.
  - `class MyRunnable implements Runnable{}`
  - Runnable is a functional interface
- Implement the abstract method run() from the interface.
  - `public void run(){}`
- Create a new instance of the Java Thread class with an instance of your class that implements Runnable as a parameter.
  - `MyRunnable mr = new MyRunnable();`
  - `Thread myThread = new Thread(mr);`
- Start the thread.
  - `myThread.start()`
  - will call the run() method in your MyRunnable instance

Benefits of implementing the Runnable interface

- Allows you to extend another class
- This is more flexible as it separates the runnable task from the Thread object that executes the task
- It is more applicable to higher level thread management APIs
  - Executor interface
  - ExecutorService interface
  - Callable objects and Future objects
  - Thread Pools