3BB4 Tutorial # 1

Greg Barkans

McMaster University

barkang@mcmaster.ca

September 20, 2016

Overview

Administrative

Regarding tutorials for this class:

- Non-Mandatory **BUT...**
  - Assignment-specific instructions
  - Assignment help
  - Participation is greatly encouraged (see Course Outline, Additional Statements)
- Bring a laptop if possible
- TAs may include their own examples or short demos that are useful for assignments

Threads and Processes

- Threads are a type of *process*
  - stand alone (sub)program
  - may execute concurrently

Threads in Java

There are 2 ways to create threads

1. ```
   class MyThread extends Thread {
       public void run() {}
   }
   ```

2. ```
   class MyRun implements Runnable {
       public void run() {}
   }
   ```

Question

Why might we need method # 2 ?

Instantiating Threads

```
    public static void main(String[] args) {
        MyThread task1 = new MyThread();
        Thread task2 = new Thread(new MyRun());
    }
}
```

Common Thread Methods

- start()
- sleep(ms)
- isAlive()
- join() $\rightarrow$ allows a thread to wait for the completion of another
  - t.join() causes current thread to pause until t finishes
  - May specify a time (ie t.join(2000)) to wait for termination

Counter Example

```
public class MyTimer extends Thread {
  // Fields
  private int time;
  final static int N = 10;

  // Constructors
  public MyTimer() {
    this.time = 0;
  }

  public MyTimer(int sec) {
    if (sec < N) {this.time = sec;}
    else {this.time = 0;}
  }
```

Counter Example

```
// Public Methods
public void run() {
  while(true) {
    System.out.println(time);
    if (time < N) {time++;}
    else if (time == N) {
      beep();
      return;
    }
    else {
      return;
    }
  }
```

Assignment # 1

Implementation details:

- use println()
- 2AA4 principles:
  - modules $\rightarrow$ do not have 1 file with nested classes
  - minimize coupling
  - commenting & style
- may run infinitely
- **implement a short delay between prints**

Special Instructions

- make and run scripts
  - if doesn't make/compile, 0
- Submit a zip with following directory structure
  - Root directory: sid_a1
  - each question should have its own sub-directory (q1 ... q4)
  - All FSP models in q4 subdirectory
  - TA must be able to: cd into directory, type 'make' then 'run'
  - Can submit answers to 1b & 2b as a txt or pdf in the proper directory

References

- J. Magee and J. Kramer. 2000. Concurrency: State Models & Java Programs, Chapters 1-2.
- `docs.oracle.com/javase/8/docs/api/java/lang/Thread.html`
- `docs.oracle.com/javase/tutorial/essential/concurrency/index.html`
- `http://www.cas.mcmaster.ca/~franek/courses/cs2xa3/`

The End