SFWRENG 3BB4 - Tutorial 1
Introduction to Java Threads

Duy Vu

McMaster University

September 24, 2016

Outline

Outline

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

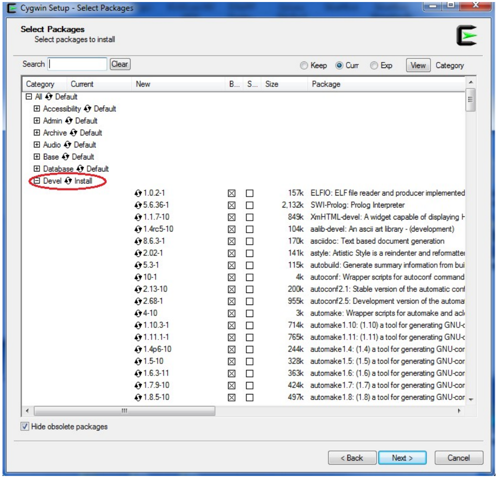
Outline

Tools

- Java IDE: Eclipse ONLY FOR WRITING CODE, NOT COMPILING OR EXECUTING APPLICATION
- GNU tools:
  - automake for compiling source codes
  - Bash shell for executing application

* Cygwin for windows, Brew for Mac

Tools



Cygwin Setup - Select Packages

Select Packages
Select packages to install

Search [ ] Clear   Keep [✓ Curr] Exp   View   Category

Category	Current	New	B...	S...	Size	Package
⊟ All ⟳ Default						
⊞ Accessibility ⟳ Default						
⊞ Admin ⟳ Default						
⊞ Archive ⟳ Default						
⊞ Audio ⟳ Default						
⊞ Base ⟳ Default						
⊞ Database ⟳ Default						
⊟ Devel ⟳ Install						
		⟳ 1.0.2-1	☒	☐	157k	ELFIO: ELF file reader and producer implemented
		⟳ 5.6.36-1	☒	☐	2,132k	SWI-Prolog: Prolog Interpreter
		⟳ 1.1.7-10	☒	☐	849k	XmHTML-devel: A widget capable of displaying H
		⟳ 1.4rc5-10	☒	☐	104k	aalib-devel: An ascii art library - (development)
		⟳ 8.6.3-1	☒	☐	170k	asciidoc: Text based document generation
		⟳ 2.02-1	☒	☐	141k	astyle: Artistic Style is a reindenter and reformatte
		⟳ 5.3-1	☒	☐	115k	autobuild: Generate summary information from bui
		⟳ 10-1	☒	☐	4k	autoconf: Wrapper scripts for autoconf command
		⟳ 2.13-10	☒	☐	200k	autoconf2.1: Stable version of the automatic conf
		⟳ 2.68-1	☒	☐	955k	autoconf2.5: Development version of the automa
		⟳ 4-10	☒	☐	3k	automake: Wrapper scripts for automake and acl
		⟳ 1.10.3-1	☒	☐	714k	automake1.10: (1.10) a tool for generating GNU-c
		⟳ 1.11.1-1	☒	☐	765k	automake1.11: (1.11) a tool for generating GNU-c
		⟳ 1.4p6-10	☒	☐	244k	automake1.4: (1.4) a tool for generating GNU-cor
		⟳ 1.5-10	☒	☐	328k	automake1.5: (1.5) a tool for generating GNU-cor
		⟳ 1.6.3-11	☒	☐	363k	automake1.6: (1.6) a tool for generating GNU-cor
		⟳ 1.7.9-10	☒	☐	424k	automake1.7: (1.7) a tool for generating GNU-cor
		⟳ 1.8.5-10	☒	☐	497k	automake1.8: (1.8) a tool for generating GNU-cor

☑ Hide obsolete packages

< Back   Next >   Cancel

source: http://stackoverflow.com/questions/4828388/cygwin-make-bash-command-not-found

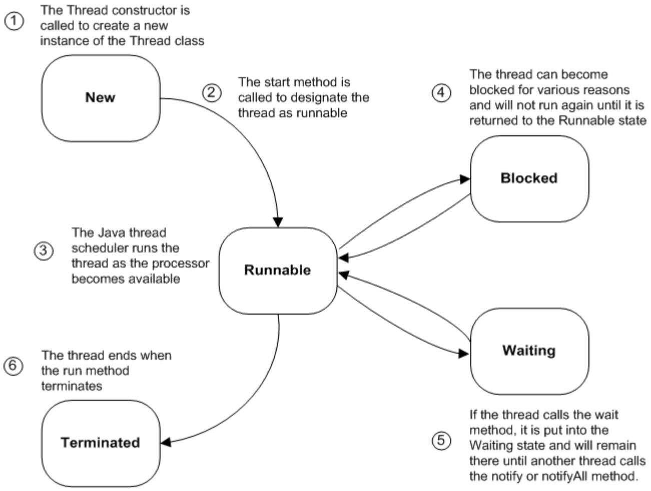
Tools

Take your own risk while using other tools

Outline

Java Threads

Java thread's life cycle


① The Thread constructor is called to create a new instance of the Thread class
New
② The start method is called to designate the thread as runnable
④ The thread can become blocked for various reasons and will not run again until it is returned to the Runnable state
Blocked
③ The Java thread scheduler runs the thread as the processor becomes available
Runnable
Waiting
⑥ The thread ends when the run method terminates
Terminated
⑤ If the thread calls the wait method, it is put into the Waiting state and will remain there until another thread calls the notify or notifyAll method.

source: http://www.c-jump.com/bcc/c257c/Week13/Week13.html

Java Threads

- Popular constructor: Thread(), Thread(Runnable target)
- Popular methods: start(), run(), join(), sleep(long millis)
- Ref: https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html

Thread class

A procedure for creating a thread

1. Create a class that inherits the Thread class
2. Override the run() method
3. Instantiate an object from the class
4. Call the start method of the thread object

Step 3 and 4 are done in the “main” method.

Thread class

Example 1: Create a thread that repeatedly and randomly prints ‘a’ twice or ‘b’ once

Runnable Interface

A procedure for creating a thread:

1. Create a class that implements the Runnable interface
2. Implement the run() method
3. Create an instance of the Runnable class.
4. Create the thread by supplying the instance of the Runnable class to the Thread constructor
5. Call the start() method of the thread object

Step 3, 4 and 5 are done in the “main” method.

Runnable Interface

Example 2: Create a thread that repeatedly and randomly prints ‘c’ twice or ‘d’ once

Questions

What are the differences between:

- a thread vs. a process
- extending Thread class and implementing Runnable interface

Outline

Makefile

Advantages:

- Simplify the compilation process
- Only re-compile things which are necessary.

Usage:

- make %Execute commands in the “default” section of the Makefile
- make target %Execute commands in the “target” section of the Makefile

Makefile

Makefile:

- Declare and assign variables: varName = value
- Reference to an variable: $(varName)
- Declare a target section:

```
target:
  <tab-not-spaces> command1
  <tab-not-sapces> command2
```

Makefile

Makefile:

- Some special targets:
  - default: is invoked by default
  - .SUFFIXES: is used to delete and define suffix list.
  - clean: contains commands to remove object, application ... files
  - suffix replacement (e.g: .java.class): builds .class files from .java files

Ref:

https://www.cs.swarthmore.edu/ newhall/unixhelp/javamakefiles.html

Makefile

Example: Create a Makefile that has

- JC variable which reference to Java compiler
- suffice replacement target to build Java files (you have to define a suffix list containing .java and .class)
- clean target to remove all class files
- default target to build Example1.java and Main.java

Outline

Bash script

A bash script:

- is a set of command lines
- starts with location of the shell (e.g: #!/bin/bash)

Example: create a script to

- dump the content of PATH variable
- run example 2 project

Outline

Assignment 1

- Submit via dropbox in Avenue.
- File name: studentId_a1.zip
- Directory structure:

```
studentId_a1
├── q1
│   ├── *.java
│   ├── Makefile
│   └── run.sh
├── ...
└── q4
    └── *.lts
```

Assignment 1

Marking scheme:

- Black box: 50%
- White box: 50%
    - Good variable, constant and method names
    - Good comments
    - No duplicated codes

Assignment 1

Why are comments important?

- Example 3:

```
int a = 10;
// What am I checking?
if ((a & 1) == 0) {
  System.out.println("\"a\"␣is␣...");
} else {
  System.out.println("\"a\"␣is␣...");
}
```

Assignment 1

Why are comments important?

- Example 4:

```
int a = 10;
// What am I checking?
if ((a & (a - 1)) == 0) {
  System.out.println("\"a\"␣is␣a␣...");
} else {
  System.out.println("\"a\"␣is␣not␣...");
}
```

Assignment 1

Remove duplicated codes by “Refactor" options in Eclipse

Assignment 1

Questions?