CISC 3115 TY3
C29a: Graphical User Interface

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Overview of user interface
    - Command Line Interface (CLI) and Graphical User Interface (GUI)
    - Comparison of CLI and GUI
- Overview of event-driven programming
    - Concept of event, event loop, event queue, and event processing/handling
    - Event handling timing requirement

User Interface

- Interface
  - A system that allows two or more entities to exchange data
- User interface
  - Typical entities are computers and humans
  - It includes both hardware and software
- Program interface
  - Typical entities are two “computer programs” (or program components)
  - What does “API” stand for?

Types of User Interfaces

- Text-based user interface
  - Often called command-line interface
- Graphical user interface

Text-based User Interface: "javac" Example

- We use "javac" to compile Java programs
- Type "javac" on the command line

```
$ javac
Usage: javac <options> <source files>
where possible options include:
  -g                        Generate all debugging info
  -g:none                   Generate no debugging info
  -g:{lines,vars,source}    Generate only some debugging info
  -nowarn                   Generate no warnings
  -verbose                  Output messages about what the compiler is doing
  -deprecation              Output source locations where deprecated APIs are used
  …..
```

Interfacing with “javac”

- Display version of “javac”

```
$javac -version

javac 1.8.0_131
```

- Compile a Java program targeting at Java version 8 or newer

```
$javac -target 8 HelloWorld.java
```

Text-based User Interface: Advantage

- Relies primarily on the keyboard and the terminal
  - Easy to customize options
  - Can do powerful tasks
  - Relatively easy to build
  - Require few resources (processor and memory) to support the interface

Text-based User Interface: Disadvantage

- Cognitively, relies heavily on user's recall rather than recognition
- Navigation is often more difficult

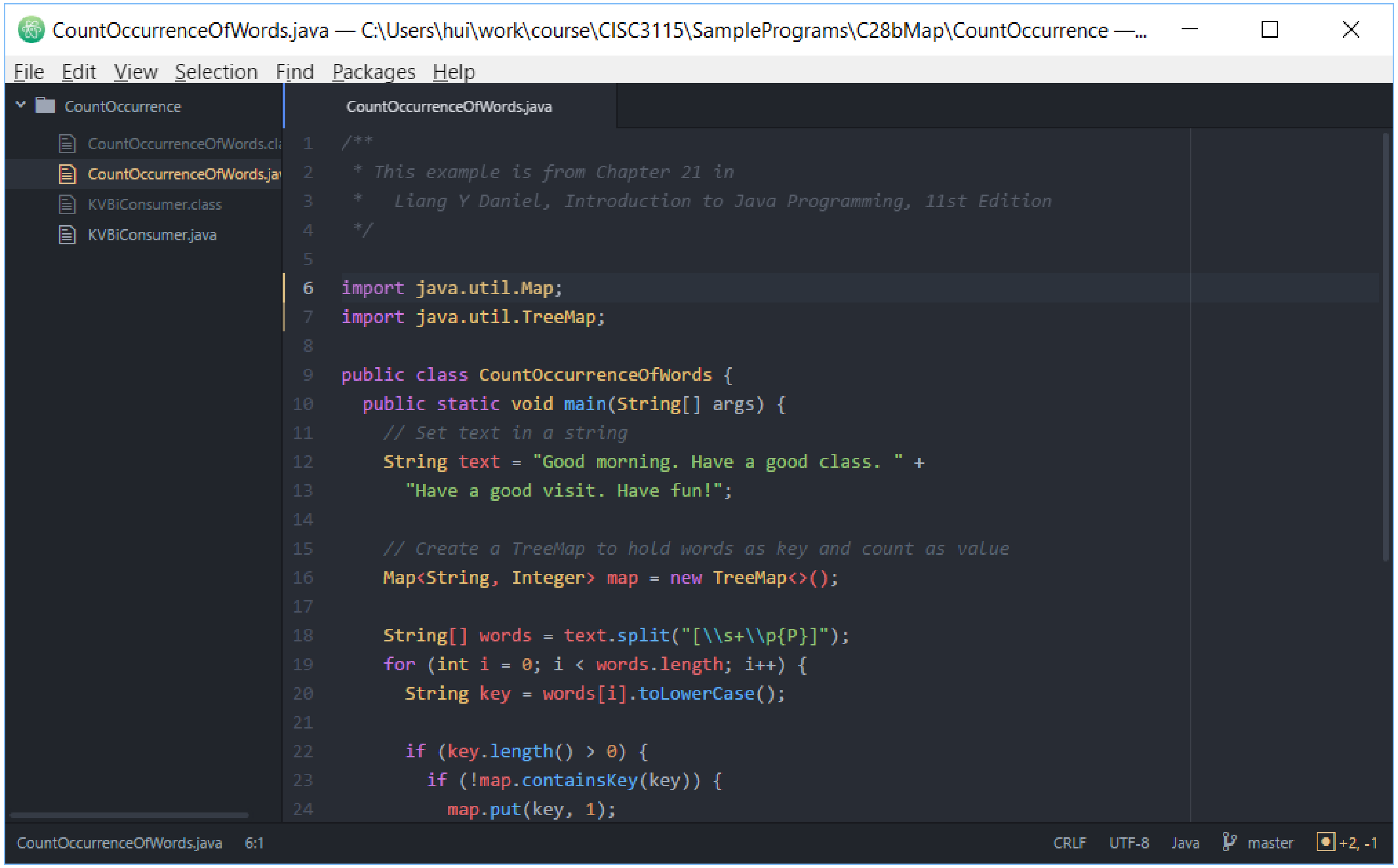
Text-based User Interface: “ls” Example

- We can use “ls” to list files on a Unix/Unix-like operating systems (Linux, Mac OS X, etc.)
  - ls –l: list files and directories in long format
  - ls –F: append character to indicate file types
  - ls –l –F: list files and directories in long format and append character to indicate file types
  - Common combinations of options is 100+

Interfacing with “ls”

- Common combinations of options is 100+
- Either frequently look up them from the user’s manual or memorize them (recall other than recognition)
- Perhaps, we can create a program that has a menu or a list buttons
  - You need 100+ menu entries or buttons

Graphical User Interface: "atom" Example


CountOccurrenceOfWords.java — C:\Users\hui\work\course\CISC3115\SamplePrograms\C28bMap\CountOccurrence —...
File Edit View Selection Find Packages Help
CountOccurrence
CountOccurrenceOfWords.cla
CountOccurrenceOfWords.ja
KVBiConsumer.class
KVBiConsumer.java
CountOccurrenceOfWords.java
CountOccurrenceOfWords.java 6:1
CRLF UTF-8 Java master +2, -1

```java
/**
 * This example is from Chapter 21 in
 *   Liang Y Daniel, Introduction to Java Programming, 11st Edition
 */

import java.util.Map;
import java.util.TreeMap;

public class CountOccurrenceOfWords {
  public static void main(String[] args) {
    // Set text in a string
    String text = "Good morning. Have a good class. " +
      "Have a good visit. Have fun!";

    // Create a TreeMap to hold words as key and count as value
    Map<String, Integer> map = new TreeMap<>();

    String[] words = text.split("[\\s+\\p{P}]");
    for (int i = 0; i < words.length; i++) {
      String key = words[i].toLowerCase();

      if (key.length() > 0) {
        if (!map.containsKey(key)) {
          map.put(key, 1);
```

Graphical User Interface

- Often use acronym: GUI
- Visualizes data for users graphically
- Often equipped with mouse, trackball, or touch pad

Graphical User Interface: Advantage

- It is often said that it provides a friendly interface between user and program
- But why?
  - Cognition
    - Cognitively, relies more on recognition than recall (less knowledge to use the application)
  - Navigation
    - Often equipped with point-and-click devices (mouse, trackball, joystick, touchpad …)
    - Allows user navigate easily

Graphical User Interface: Disadvantage

- Typically decreased options (less powerful) when compared to command line interface
- Typically less customizable..
  - Recall the “ls” example
  - Not easy to express many combinations of options in GUI
  - Not easy to use one set of button for many different options or combinations in GUI

Questions

- Text-based user interface
- Graphical user interface
- Disadvantage and advantage

GUI and Event-Driven Programming

- More user friendly and easy navigation
- GUI applications are popular in modern computing
- Allows <u>event-driven or reactive programming</u>
- Often multi-threaded: allows multiple concurrent threads of executions

Event

- An event is an object (created from an event source) that drives the execution of a program
- Semantics of event
    - Serves as a type of signal to the program that something has happened.
- Example events
    - Generated by external user actions such as mouse movements, mouse clicks, or keystrokes
    - A mouse click
    - A keyboard stroke
    - A timeout of a timer
    - A window is closed

Event-Driven Programming

- The main body of the program is an event loop (in pseudo code)

```
do {
    e = getNextEvent()
    processEvent(e)
} while (e != EXIT_EVENT)
```

$\Bigg\}$ Expected to be completed very quickly (a fraction of a second)

- This event loop often implemented by the platform
- The events are in a queue called event queue (capacity? priority?)
- Users write event handler routines (user's programs) to process events
  - processEvent in the above will invoke your event handler routines
  - Event thus drives user's programs

Event-Driven and Algorithm-Driven (Application-Driven)

- Application-driven or algorithm-driven programs
  - A program expects inputs in a pre-determined order and timing
  - e.g., Project 1 and Project 2
- Event-driven programming
  - Program waits for input events when it loads
  - The programs runs particular code to response with an event
  - The overall flow of the execution is determined by the events that occur
  - The overall flow of what code is executed is determined by events in non-deterministic order and timing
  - A type of reactive programming

Questions?

- Concept of event-driven programming
- Comparison of event-driven programming and algorithm-driven programming (application-driven or procedure-driven)
- Concept of event, event handler, event loop, and event queue
- Timing requirement for processing events