COMP90041
Programming and Software Development
2020 - Semester 1
Lab 10 - Week The Last

Yuhao(Beacon) Song

yuhsong1@unimelb.edu.au

Introduction

- Timetable
  - Tue(11) 14:15-15:15 (Melbourne Time) Join URL: https://unimelb.zoom.us/j/490084146
  - Tue(07) 16:15-17.15 (Melbourne Time) Join URL: https://unimelb.zoom.us/j/291505765

- Contact
  - yuhsong1@unimelb.edu.au
  - yuhsong@student.unimelb.edu.au
  - GitHub: https://github.com/Beaconsyh08/COMP90041_Programming_and_Software_Development_Tutorials.git

Outline

- Javadoc
- Command-Line Arguments
- Random Seed
- Others

Javadoc

- Unlike a language such as C++, Java places both the interface and the implementation of a class in the same file
- However, Java has a program called **javadoc** that automatically extracts the interface from a class definition and produces **documentation**
  - This information is presented in HTML format, and can be viewed with a Web browser
  - If a class is correctly commented, a programmer need only refer to this API (Application Programming Interface) documentation in order to use the class
  - **javadoc** can obtain documentation for anything from a single class to an entire package

- https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html

Javadoc Comment

- The javadoc program extracts class headings, the headings for some comments, and headings for all public methods, instance variables, and static variables
  - In the normal default mode, no method bodies or private items are extracted
- To **extract a comment**, the following must be true:
  - The comment must immediately precede a public class or method definition, or some other public item
  - The comment must be a block comment, and the opening /*must contain an extra * **(/** . . . */)**
  - Note: **Extra options** would have to be set in order to extract line comments (//) and private items
- In addition to any general information, the comment preceding a public method definition should include descriptions of **parameters**, any value **returned**, and any **exceptions** that might be thrown
  - This type of information is preceded by the **@ symbol** and is called an **@ tag**
  - **@ tags** come after any general comment, and each one is on a line by itself

@ Tags

- **@ tags** should be placed in the order found below
- If there are multiple items, each should have its own @xxx on a separate line, and each should be listed according to its left-to-right order if they have

- ***@param*** *Parameter_Name Parameter_Description*
- ***@return*** *Description_Of_Value_Returned*
- ***@throws*** *Exception_Type Explanation*
- ***@author*** *Author*
- ***@version*** *Version_Information*

Javadoc Comment For Class

```java
/**
 * Description
 * The NimPlayer is an Abstract class
 * This class includes the basic information, getter and setter method about player
 * This class has an abstract method removeStone, which will be implement in it's subclasses.
 *
 * @author YUHAO SONG
 * @version 1.3
 */
public abstract class NimPlayer {
```

Javadoc Comment For Method

```java
/**
 * Allows new players to be added to the game.
 *
 * @param userAmount  total amount of user
 * @param command     cmd
 * @param playerArray Nimplayer array
 * @return the number of userAmount
 * @throws InvalidNumberOfArgumentsException amount of arguments not equal to expected
 */
private int addPlayer(int userAmount, String[] command,
                      NimPlayer[] playerArray) throws InvalidNumberOfArgumentsException {
```

Command-Line Arguments

- A Java application can accept any number of arguments from the **command line**. This allows the user to specify configuration information when the application is launched.

- The user enters command-line arguments when invoking the application and specifies them after the name of the class to be run.

- When an application is launched, the runtime system passes the command-line arguments to the application‘s **main method** via an **array of Strings → args.**

```
public static void main(String[] args) {
```

Command-Line Arguments Conventions

- The **dash character (-)** precedes options, **flags**, or series of flags.
- Arguments can be given in any order, except where an argument requires another argument.
- **Flags** can be listed in any order, separately or combined: -xn or -nx or -x -n.
- Filenames typically come last.
- The program prints a usage error when a command line argument is unrecognized. Usage statements usually take the form:

  usage: *application_name* [optional_args] required_args

- http://journals.ecs.soton.ac.uk/java/tutorial/java/cmdLineArgs/parsing.html

Parsing Numeric Command-Line Arguments

- If an application needs to support a **numeric** command-line argument, it **must convert a String argument** that represents a number, such as "34", to a **numeric** value.

- **parseInt** throws a **NumberFormatException** if the format of args[i] isn't valid. All of the Number classes — Integer, Float, Double, and so on — have **parseXXX** methods that convert a String representing a number to an object of their type.

- https://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html

Random Seed

- A **random seed** (or seed state, or just seed) is a number (or vector) used to initialize a pseudorandom number generator.
- For a seed to be used in a pseudorandom number generator, it does not need to be random. Because of the nature of number generating algorithms, so long as the original seed is ignored, the rest of the values that the algorithm generates will follow probability distribution in a pseudorandom manner.
- A pseudorandom number generator's number sequence is completely **determined** by the seed: thus, if a pseudorandom number generator is reinitialized with the same seed, it will produce the **same** sequence of numbers.

- [Pseudorandom number generators | Computer Science | Khan Academy] https://www.youtube.com/watch?v=GtOt7EBNEwQ&t=1s

Random Seed for Computer

- A **random seed** specifies the **start point when a computer generates a random number sequence.** This can be any number, but it usually comes from seconds on a computer system's clock (Henkemans & Lee, 2001)
- → ***Unix Time:*** January 1st, 1970 ~ January 19, 2038 UTC
- On this date the Unix Time Stamp will cease to work due to a 32-bit overflow. Before this moment millions of applications will need to either adopt a new convention for time stamps or be migrated to 64-bit systems which will buy the time stamp a "bit" more time.
- [Current Unix Time] https://www.epochconverter.com/

Some Methods for Random in Java

`setSeed(long seed)`

Sets the seed of this random number generator using a single `long` seed.

`nextInt()`

Returns the next pseudorandom, uniformly distributed `int` value from this random number generator's sequence.

`nextInt(int bound)`

Returns a pseudorandom, uniformly distributed `int` value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

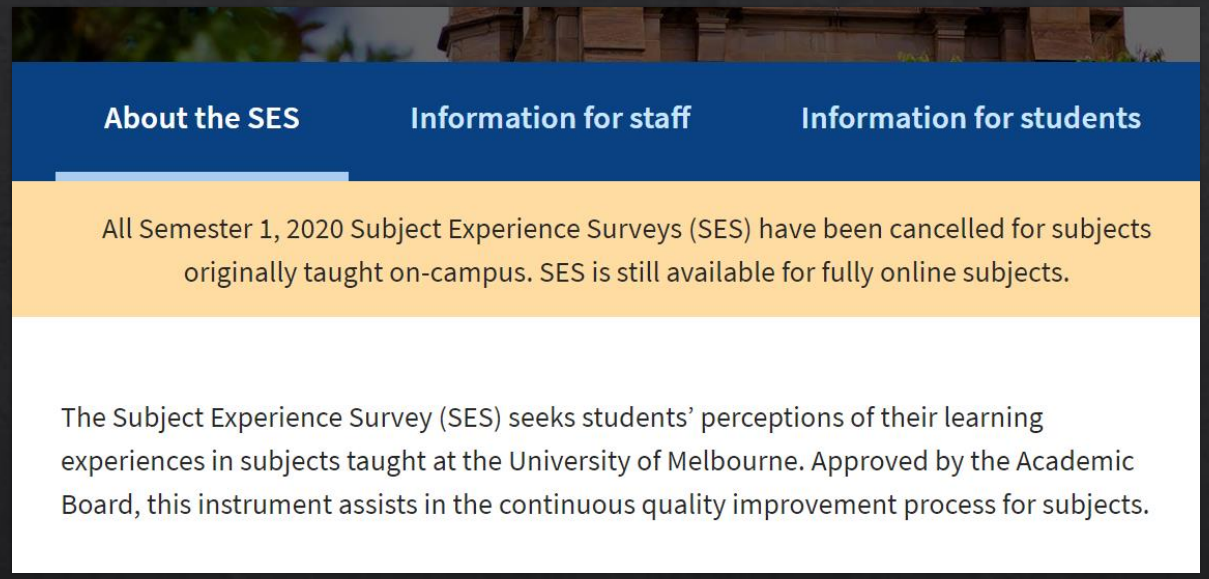
- https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

Why Seed

- Computers don't generate truly random numbers—they are **deterministic**, which means that they operate by a set of rules. (Just toy example, **much much more complicated** in reality)
  - For example, "take a number x, add 900, then subtract 52." In order for the process to start, you have to specify a starting number, x (the seed). Let's take the starting number 77:
    - 77 + 900 – 52 = 925
  - Following the same algorithm, the second "random" number would be:
    - 925 + 900 - 52 = 1773

- For example, let's say you wanted to generate a random number. If you set a number for **Seed**, you'll be able to use the same output of random numbers again. If you set 22 this time, and set 22 for the next time you run the random number generator, you will get the **same** number. If you set 33, you'll get an entirely **different** number.

Survey



- No official SES this semester.
- However, Your feedback is important for me ☺
- https://forms.gle/gUHDjYAKUTbV547m8

ANYTHING ABOUT THIS SUBJECT. PLEASE USE EMAIL OR DISCUSSION BOARD ONLY!!!

- Ins

- LinkedIn

- GitHub Repo

- WeChat

- Email
  - yuhsong1@unimelb.edu.au
  - yuhsong@student.unimelb.edu.au
- WhatsApp
  - +61 435 333 386

◈ Thank you!!!