Welcome to cs351 - 801 (Fall-2019)

Seyedamirhossein Hesamian (Amir)

- Please ask questions on:
  - piazza.com
- Please send email questions to:
  - compsci-351@uwm.edu
- Personal email:
  - hesamian@uwm.edu
- Office hour: Friday 3:30-4:30
- Room: PHY-232
- Slides: github.com/amir734jj/cs351

Syllabus

- Lab grades:

Grade	Description
0	Nothing has been done
IP	Lab assignment is not complete
P	Complete lab assignment

- 10% of total grade
  - One lab will be dropped
  - *24 hour* after the lab to submit the lab

"cs351 is *difficult* course but a *rewarding*"

(Quote from syllabus)

It is an axiom that data structures and algorithms are the base of computer science. It is the fundamental of computer science.

Furthermore:

- Technical interviews
- Daily tasks as a software developer
- Career as a computer scientist

P.S.: I am a UWM alumni class of 2015; I took cs351 in Spring 2013!

ADT vs. Data Structure

- Description:
  - **ADT** is a *logical* description and **data structure** is *concrete*.
  - **ADT** is the *logical picture* of the data and the operations to manipulate the component elements of the data. **Data structure** is the *actual representation* of the data during the implementation and the algorithms to manipulate the data elements.
  - **ADT** is in the *logical level* and **data structure** is in the *implementation level*.

ADT vs. Data Structure (Cont'd)

- Example:

ADT	Data Structure
List	ArrayList, LinkedList
Set	SortedSet, HashSet, TreeSet, LinkedHashSet
Map	SortedMap, HashMap, TreeMap LinkedHashMap

Time Complexity

Notation:

- Big O (O) gives upper bound (**most commonly used**)
- Big Omega (Ω) gives lower bound and
- Big Theta (Θ) gives both lower and upper bounds

Time Complexity (Cont'd)

```
// Do something ...

System.out.println("Welcome to cs351");
```

$\rightarrow O(1)$ **or constant time**

Time Complexity (Cont'd)

```
int[] arr = ...
int n = arr.length;

for (int i = 0; i < n; i++) {
  // Do something ...
}
```

$\rightarrow O(n)$ or *Linear* time

Time Complexity (Cont'd)

```
int n = ...

for (int i = 1; i < n; i = i * 2) {
  // Do something ...
}
```

$\rightarrow O(n * log(n))$ or ***Logarithmic*** time

Time Complexity (Cont'd)

```
int n = ...
int[][] arr = new int[n][n];

for (int i = 0; i < n; i++) {
  for (int j = 0; j < n; j++) {
    // Do something ...
  }
}
```

$\rightarrow O(n^2)$ or *Polynomial* time

Time Complexity (Cont'd)

```java
public int bruteForce(String text, String pattern) {
    int length = text.length();      // length of the text
    int plength = pattern.length(); // length of the pattern

    for (int i = 0; i < length - plength; i++) {
        int j = 0;

        while ((j < plength) && (text.charAt(i + j) == pattern.charAt(j))) {
            j++;
        }

        if (j == plength) { return i; }
    }

    return -1;
}

// bruteForce("Hello World!", "World") => ?
```

$\rightarrow O(2^n)$ or *Exponential* time

Time Complexity (Cont'd)

- In upcoming homeworks, we will have *efficiency tests*!
  - Writing a code that works may not be enough
  - Code has to be efficient, avoid exponential time complexities

Debugging

- Notation
  - Step ***over***
  - Step ***into***
  - Step ***out***

Debugging (Cont'd)

```
void Foo(int n) {
  int k = n / 2;
  System.out.println(k);
}

void Bar(int n) {
  int k = n / 3;
  System.out.println(k);
}

void Baz(int n) {
  int k = n / 5;
  System.out.println(k);
}

Foo();              // <-- debugger breakpoint is here!
Bar();
Baz();
```

`git`

Git is a distributed version-control system for tracking changes in source code during software development.

- Try to at least get familiar with it
- Essential for your career as a software developer and this course

`git` (Cont'd)

Useful Git commands:

Command	Example
`clone`	`git clone <url> .`
`add`	`git add .`
`commit`	`git commit -m "started working on HW1"`
`push`	`git push`
`pull`	`git pull`
`log`	`git log`

`git` (Cont'd)

```
clone     // Clone the repository (or repo) locally

while !finished_assignment():
  add     // Stage your changes to be committed
  commit  // Commit your changes
  push    // Push your commit(s) to remote (i.e. afs)
```

`git` command line

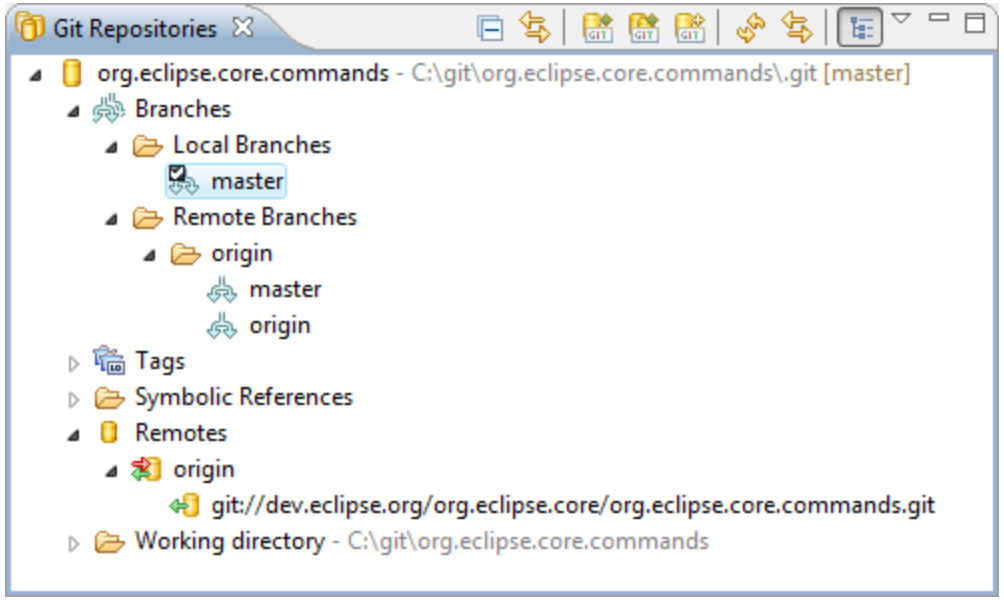
More powerful but steep learning curve

MINGW32:~/git

```
Welcome to Git (version 1.8.3-preview20130601)


Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Bacon@BACON ~
$ git clone https://github.com/msysgit/git.git
Cloning into 'git'...
remote: Counting objects: 177468, done.
remote: Compressing objects: 100% (52057/52057), done.
remote: Total 177468 (delta 133396), reused 166093 (delta 123576)
Receiving objects: 100% (177468/177468), 42.16 MiB | 1.84 MiB/s, done.
Resolving deltas: 100% (133396/133396), done.
Checking out files: 100% (2576/2576), done.

Bacon@BACON ~
$ cd git

Bacon@BACON ~/git (master)
$ git status
# On branch master
nothing to commit, working directory clean

Bacon@BACON ~/git (master)
$
```

EGit

Eclipse EGit plugin is recommended for this class. EGit is a GUI for `git`



Lab assignment #1: