Computational Thinking
Computer Science and Programming

Computer Science

- Broad discipline that explores any and all areas of computation.
- Includes: theory of computation, algorithms, computer graphics, language theory, systems, and human-computer interaction.

Programming

- Providing a set of instructions to a computer to automate a specific task or solve a given problem.
- An application of computer science.
- Implementing algorithms.
Problem Solving
Art of Programming

Abstraction
- Can you see the essential parts of the problem?
- What are the outputs? inputs? their relationship?
- Can you summerize the problem at a high level?
- What are the different components of the solution?
Problem Solving
Art of Programming

Abstraction
- Can you see the essential parts of the problem?
- What are the outputs? inputs? their relationship?
- Can you summarize the problem at a high level?
- What are the different components of the solution?

Computational Thinking
- Think like a computer.
- What is the sequence of actions need to accomplish the task?
- Start with *pseudocode*. 
Exercise

Calculating the area and circumference of a circle.

1. What are the inputs? outputs? their relationship?
2. Pseudocode it!
3. Code it!
Computer
Basic Machine Architecture

von Neumann Architecture

- von Neumann proposed this architecture in 1945.
- Consists of:
  - a processing unit,
  - memory,
  - input devices, and
  - output devices.

“This is a computer.”
**Basic Machine Architecture**

von Neumann Architecture

- von Neumann proposed this architecture in 1945.
- Consists of:
  - a processing unit,
  - memory,
  - input devices, and
  - output devices.

John von Neumann
Basic View of a Computer
Explaining and Tracing
Explaining vs Tracing

Explaining
Summerize and provide a high-level explanation of what the code does in plain English.
Explaining vs Tracing

Explaining
Summerize and provide a high-level explanation of what the code does in plain English.

Tracing
Run the code as computer does.
- Put pen to paper.
- Write down the active variables and their values.
- Update them as they change as you mentally walk through the statements sequentially.
**TopHat Q4**

**Explain the `Circle.java` code:**

a. Calculates the area and the circumference of a circle.

b. Reads a radius as input from the user and outputs the area and the circumference of the corresponding circle.

c. Creates a double variable called `rad`. Initializes it as 0. Prompts the user for a radius and stores it in `rad`. Outputs “Area:”; calculates the area of a circle of radius `rad` and outputs it. Outputs “Circumference:”; calculates the circumference of a circle of radius `rad` and outputs it.
**Tool for Tracing**

**Java Visualizer**

- [https://cscircles.cemc.uwaterloo.ca/java_visualize/](https://cscircles.cemc.uwaterloo.ca/java_visualize/)
Edit-Compile-Run Cycle
Edit

Writing some source code (set of instructions) in plain text.

Editors

- Text Editors:
  - vi
  - Vim
Writing some source code (set of instructions) in plain text.

**Editors**

- **Text Editors:** vi, Vim
- **Integrated Development Environment (IDE):** Eclipse, Visual Studio
Compile

Build your code so that it can be run on a computer. Often compilers build an *executable* that can be run *natively* on the *target platform*.
Compile

Build your code so that it can be run on a computer. Often compilers build an *executable* that can be run *natively* on the *target platform*.

**javac – The *Java* compiler**

- javac produces a *bytecode* file that needs to be run in a *virtual machine*. 
Compile

Build your code so that it can be run on a computer. Often compilers build an executable that can be run natively on the target platform.

javac – The Java compiler

- javac produces a bytecode file that needs to be run in a virtual machine.
- javac produces a bytecode file with a .class file extension.
Compile

Build your code so that it can be run on a computer. Often compilers build an *executable* that can be run *natively* on the *target platform*.

**javac – The *Java* compiler**

- javac produces a *bytecode* file that needs to be run in a *virtual machine*.
- *javac* produces a bytecode file with a `.class` file extension.
- Example:
  
  ```bash
  javac Circle.java
  ```
  produces a `Circle.class` bytecode file.
Run or execute your code.

`java`

- The .class bytecode file produced by javac needs to be run in the Java virtual machine.
Run

Run or execute your code.

**java**

- The .class bytecode file produced by javac needs to be run in the Java virtual machine.
- Example:
  
  ```java
  java Circle
  ```

  runs the Circle file.
Virtual Machine

The Classic Model: Native executable

Source Code → Compilation → Machine Code (executable)

The Java Model: Virtual Machine

Source Code → Compilation javac foo.java → Bytecode foo.class → Execution java foo → Virtual Machine
I have just written some Java code in a file SomeCode.java. How do I compile and run it?

a. java SomeCode
b. javac SomeCode.class
c. javac SomeCode.java
d. java SomeCode.java
e. javac SomeCode
f. java SomeCode.class
FURTHER READING

COMP SCI 200: Programming I
zyBook code:
WISCCOMPSCI200Fall2017

● Chapter 1. Programming Process
Appendix
References
Image Sources I

http://www.eclipse.org/

https://www.gnu.org/software/emacs/emacs.html

http://packerville.blogspot.ca/2010/05/gentlemen-this-is-football.html

https://en.wikipedia.org/wiki/Microsoft_Notepad

http://2dvocabularynetwokr78.blogspot.fr/
Image Sources II

http://www.lanl.gov/history/atomicbomb/images/NeumannL.GIF

https://brand.wisc.edu/web/logos/

https://commons.wikimedia.org/w/index.php?curid=1228427

https://commons.wikimedia.org/w/index.php?curid=57649239

http://www.zybooks.com/