CST8132 OBJECT-ORIENTED PROGRAMMING

Overview & Review

Professor : Dr. Anu Thomas

Email: thomasa@algonquincollege.com

Office: T314

Today’s Topics

- Brightspace
- Course Outline
- Reflections on CST8116
- Intro to Object-Oriented Programming
- Multi-dimensional arrays
- Details on first lab: Lab 1

Brightspace

- Professor Information
  - Dr. Anu Thomas
  - T314
  - thomasa@algonquincollege.com
  - Office Hours: by appointment
- Course Information
  - Course Outline
    - What this course is supposed to be
    - Details on evaluation: test, quizzes, labs, assignments, exam & final mark
  - Course Section Information (Under Course Information)
    - A summary of what's going to happen and when throughout this term
    - Specific details on evaluation: test, quizzes, labs, assignments, exam & final mark

Brightspace (Cont'd)

- Weekly Learning Materials
  - Lecture Notes, PowerPoint slides etc.
    - before joining the class, check slides
- Hybrid Assignments
  - Materials and links that you will work on your own, with online quizzes
- Lab Assignments
  - This is the area where you will retrieve documents and related files for labs
  - Labs should be submitted before the due date and should be demoed to your lab professor during the lab sessions
  - Both submission AND demo are required to get grades
- **Survey - Location**

Brightspace (Cont’d)

- Discussion Board
    - We can use these to discuss details of lab requirements
    - Questions about intended purpose or satisfactory strategies etc.
    - There may not be much activity, but it is a resource available to us
    - Can have discussions on various course topics, and you can help each other by posting answers
    - Don’t post code for any assignments!
- Announcements
    - When a student asks a question, and if it is relevant for the entire class, I will post an announcement

Resources

- Required Textbook:
  - Java, How to Program, 11th Edition by Deitel and Deitel, Published by Pearson Education Inc. in 2015 ISBN: 9780134743356

- Recommended Textbook:
  - Effective Java, 3rd Edition, Joshua Bloch, Addison-Wesley Professional, ISBN: 9780134686097
  - Big Java Early Objects, Seventh Edition, Cay Horstmann, Wiley, ISBN: 978-1-119635659

- Software:
  - Java Platform (JDK) 8uNN (latest Java 8 update)
  - - https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
  - Eclipse (as used in CST8116)
    - http://www.eclipse.org/downloads

Teaching / Learning Methods

- Lectures
  - 2 hours per week
  - some (but not all) notes/material posted on Brightspace
  - DO NOT MISS LECTURES
  - 1 hour hybrid per week; activity will be listed in Brightspace
  - Bonus quizzes in class (from week 2 onward)
- Labs
  - 2 hours per week – DO NOT MISS LABS
  - 6 lab questions will be completed and demoed during the lab time – (that emphasize key concepts)
  - Lab requirements will be posted in Brightspace

Your weekly schedule

- Lecture - Thursdays
  - Lecture from 10:00 M – 12:00 PM
- Lab – Mondays, Tuesdays & Fridays
  - Lab submissions are due on Sundays 11:59 PM
  - Labs should be submitted in Brightspace **AND** should be demoed during lab hours
  - CSI has detailed information of labs and its due dates
- Hybrid – due on Sundays
  - Released on Mondays, due on Sundays.
  - Multiple attempts are allowed, and will IGNORE all but the most recent submission
  - For hybrids 1, 5, and 9, only 3 attempts, but you need to do the questions that you got wrong in the previous attempt
- Lecture – Dr. Anu Thomas
- Labs: Dr. James Mwangi, Karan Kalsi

Evaluation

• Midterms	20%
• Midterm 1 (10%)	
• Midterm 2 (10%)	
• Final Exam	30%
• Lab Assignments	40%
• 3 labs (5% for lab1, 7% for remaining labs)	
• Hybrid Exercises	10%
• 10 exercises worth 1% each	

- NOTE: In order to pass this course, at least 50% (i.e., 25/50) must be achieved in the theory component (midterm and final). Additionally, at least 50% (i.e., 20/40) must be achieved in the applied component (labs). Even if your combined grade exceeds 50% for the entire course, if you fail either the theory component or the applied component you will not achieve a passing grade in the course.

Plagiarism

- **The School of Advanced Technology's Standard Operating Procedure on Plagiarism and Academic Honesty** defines plagiarism as an attempt to use or pass off as one's own idea or product, work of another without giving credit. Plagiarism has occurred in instances where a student either directly copies another person's work without acknowledgement; or, closely paraphrases the equivalent of a short paragraph or more without acknowledgement; or, borrows, without acknowledgement, any ideas in a clear and recognizable form in such a way as to present them as one's own thought, where such ideas, if they were the student's own would contribute to the merit of his or her own work.

- Plagiarism is one of the most serious academic offences a student can commit. Anyone found guilty will, on the first offence, be given a written warning and an "F" on the plagiarized work. If the student commits a second offence, an "F" will be given for the course along with a written warning. A third offence will result in suspension from the program and/or the college.

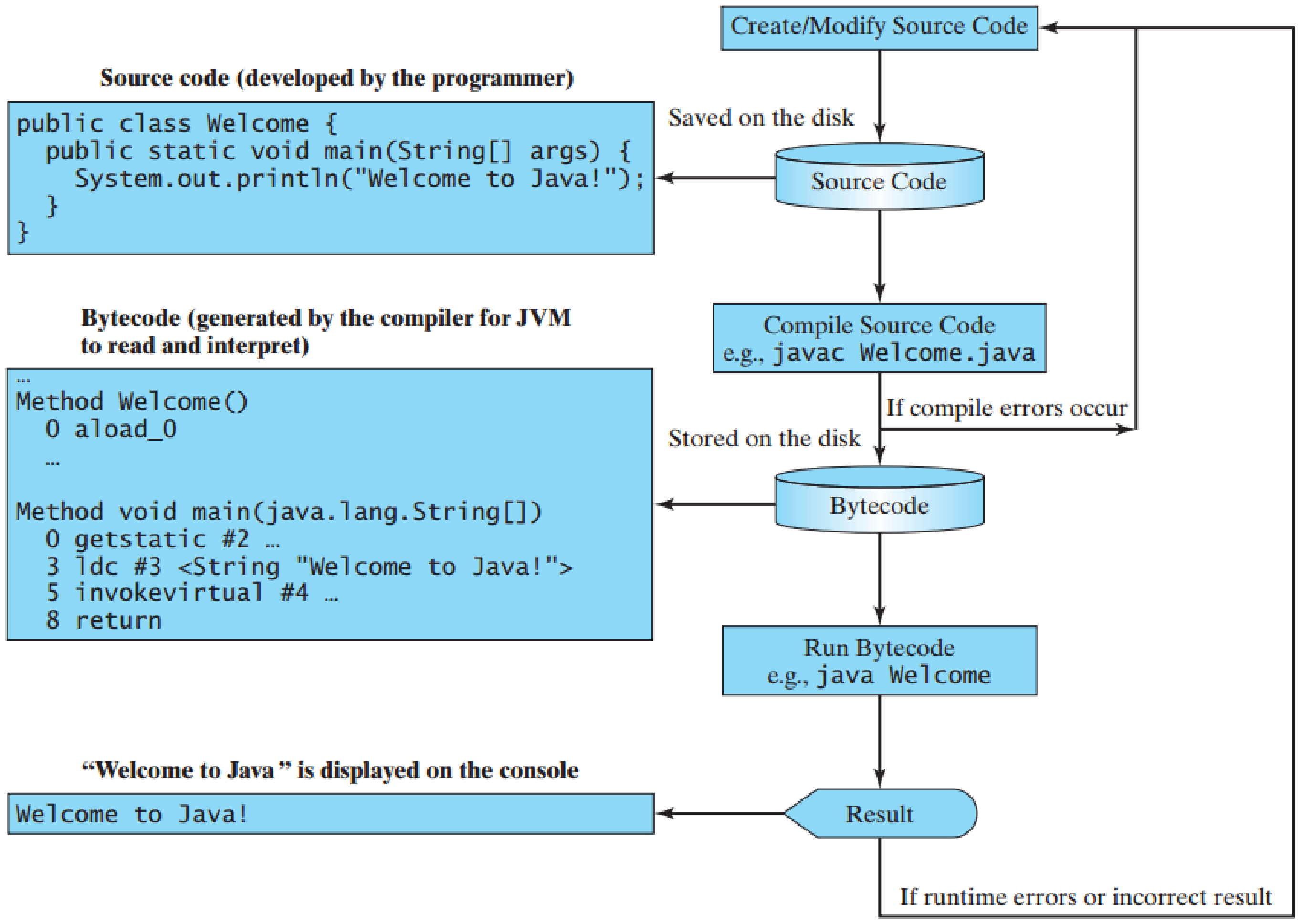
Reflections on CST8116

- Discussion on
  - What did you like about CST8116?
  - What did you dislike about CST8116?
  - Was it more difficult or easier than expected? How?
  - Was the content of CST8116 as you expected? If not, what did you expect?

CST8116 Review – JDK, JVM, JRE

- JVM – Java Virtual Machine
  - interprets Java bytecode on native hardware (HW)
    - Just In Time (JIT) compiler optimizes frequently used bytecode
  - Browser plug-in is JVM with security restrictions
- JRE – Java Runtime Environment
  - implementation of JVM, libraries & other binaries
- JDK – Java Development Kit
  - JRE plus compiler, debugger & other tools
  - JDK & JRE specific to each combination of HW &OS

CST8116 Review – JDK, JVM, JRE



Source code (developed by the programmer)

```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Bytecode (generated by the compiler for JVM to read and interpret)

```
…
Method Welcome()
  0 aload_0
  …

Method void main(java.lang.String[])
  0 getstatic #2 …
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 …
  8 return
```

"Welcome to Java " is displayed on the console

```
Welcome to Java!
```

FIGURE 1.6 The Java program-development process consists of repeatedly creating/modifying source code, compiling, and executing programs.

Development Cycle for Java

- Step 1 - Editor
  - Type in our instructions/statements according to rules of Java Language
  - File from editor is called a source file and will have extension **.java**
- Step 2 - Compiler
  - Use compiler to translate the statements in source file to bytecode (which are portable – not dependent on hardware platform)
  - File from compiler is called bytecode file and will have extension **.class**
  - Use command> `javac ProgramName.java`
- Step 3 – Class Loader
  - class loader loads bytecode files into memory (includes all the .class files that you have written/produced and those provided by Java that you are using)
  - Use command> `java ProgramName` to invoke Loader, Verification, Execution
- Step 4 – Bytecode verifier
  - Use bytecode verifier to ensure bytecodes are valid and do not violate Java's security restrictions
- Step 5 – Execution
  - Use JVM (Java Virtual Machine) to execute program's bytecodes (to perform the actions specified by your program)

CST8116 Review – Data Types

- Primitive Data Types
  - You should know sizes and literals for each

boolean	int
char	long
byte	float
short	double

- Reference Data Types
  - Object
  - String
  - Arrays

CST8116 Review – Operators

- Relational and Logical Operators
  - <   >   ==   !=   <=   >=   && ||
- Arithmetic, Compound Assignment, Increment/Decrement
  - +   -   *   /   %   +=   -=   *=   /=   %-
  - ++   --
  - Integer arithmetic (including rounding and truncation)
- Operator precedence

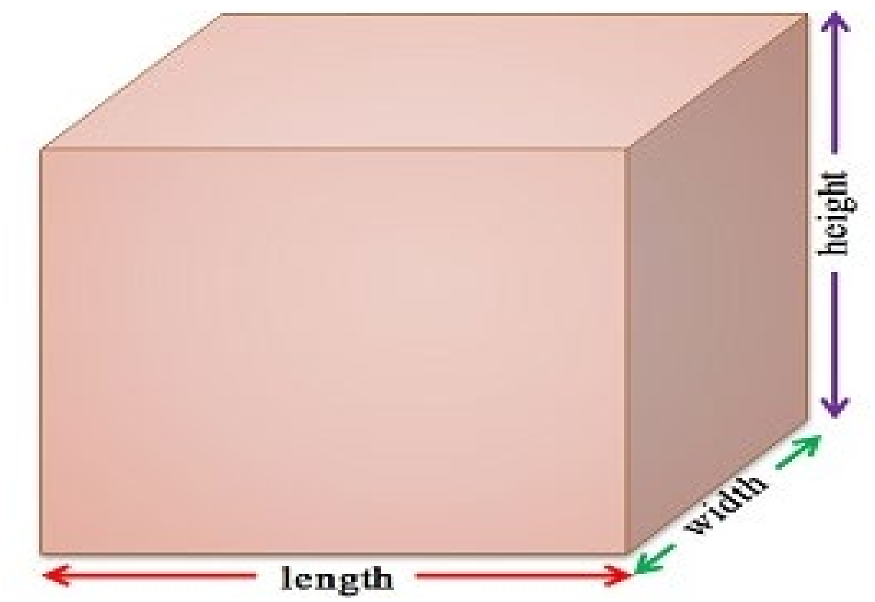
CST8116 Review – Output

- Standard Output
  - System.out.print()
  - System.out.println()
  - System.out.printf()
- Special / escaped characters
  - \n   \t   \\   \”   \’   \r
- Format specifications
  - Including width, precision, justification, and leading zeros
    - %s   %d   %x   %X   %f   etc

Object-Oriented Programming (OOP)

- In Object-Oriented programming, everything is an Object
- In Object-Oriented programming, everything **is-a**n Object
- We now have a new OOP term: **is-a**
  - More on this later
- Every object is an instance of the Object Class
- What is the difference between an object and a class?

Box



What can we do with this data?

Picture taken from:
https://keydifferences.com/wp-content/uploads/2017/01/length-vs-height.jpg

- Data (Attributes)
  - Length
  - Width
  - Height

- Find Volume?
- Find Surface Area?
- Print Volume?
- Print Surface Area?

Box of Mouse

- Length = 16 cm
- Width = 8 cm
- Height = 3 cm

- Volume = length * width * height = 384

Box of Pencils

- Length = 16 cm
- Width = 7 cm
- Height = 2.5 cm

- Volume = length * width * height = 280

Class – An example

```
public class Person{
    private int age;
    public Person(){
        age = 4;
    }
}
```

we put this in a file called Person.java

- Everything is an object
- Here, Person can have many attributes or properties like age, name, dob, weight, height etc.
- If we want to manipulate these attributes or values, we need to write methods around it
- When we wrap attributes and its corresponding methods into one single entity, that entity is called Class

Objects in OOP

- Class
  - A class is like a blueprint/drawings of a house
  - A class is the definition of a type of object
  - You cannot live in a blueprint – a class is not an object
- Object
  - An object is like the actual house
  - You can make as many houses as you want from one blueprint
    - Those houses will look the same, but there are several of them
  - Objects are created at run-time by instantiating instances of classes
  - For the class Person, we can create its object with this java code:

```
Person person1 = new Person();
```

- Instance
  - Objects of the same type are called instances of a class

Class

- We use classes in OOP to specify two things:
  1. What are the attributes of the objects we have in mind
     - ✓ The attributes of an object define its state
  2. What can those objects do (methods)
     - ✓ The methods of an object define its behavior
- So, we have learned to think of objects (Person, Object, Scanner etc) as having **state** and **behavior**
- Our Person class on the previous slide is rather silly:
  - ✓ What comprises the state of a person? As we’ve written it, just age.
  - ✓ What are the behaviors of a person? As we’ve written it, none.

public static void main(String [] args)

- The main method is a *static* method (notice the keyword static)
  - Static methods are associated with the class itself, not the object instances of the class (more on static later)
  - For our purposes right now, this means we don't need an object to exist before we call a static method
  - The main method is the starting point for Java programs
  - When the program starts, no objects exist, so main must be static
  - In CST8116, you may have done substantial programming in the main method
  - In OOP, we use the main method just to create the first objects and start them *talking* to each other

Discussion on methods

- Print age of `person1`
- Can we have a `person2`?
- Can `person2` have a different age?

Let’s implement the complete solution for review

Features of OOP

- Encapsulation
- Abstraction
- Polymorphism
- Inheritance

We will learn about these features in this term!

CST8116 Review -Control Structures

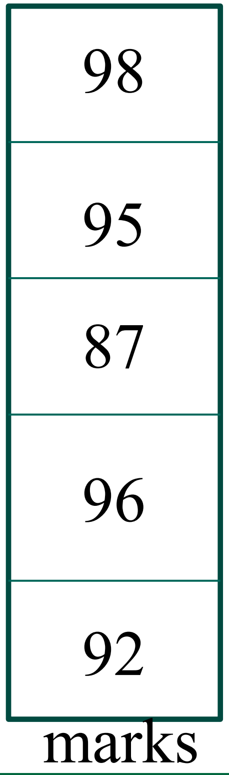
- Branching Control Structures
  - if (*condition*) { }
  - if (*condition*) { } else { }
  - if (*condition*) { } else if (*condition*) { } else { }
  - switch (*var*) { case *value*: default: }

CST8116 Review -Control Structures

- Looping Control Structures
  - while (*condition*) { }
  - do { } while (*condition*);
  - for (*initialization*; *termination*; *increment*) { }
  - for (Object o : [collection|array]) { }
    - Enhanced for loop
    - For-each loop

CST8116 Review – Arrays

- An array is a container object which holds a specified number of values of a single data type.
- The length of the array is specified when the array is initialized.
  - Once created, the length of an array is fixed.
- Each item in an array is called an **element**
- Each element is accessed by its **index**.
- Index numbering always begins with **0**.



CST8116 Review – Arrays

- An array declaration has two required parts:
  - The array's type, written as ***type*[ ]**
  - The array's name
- An array must be both declared and initialized before the elements of the array may be accessed.
- Arrays are often accessed using looping constructs.

```
int []marks= new int[5];
```

marks	
marks[0]	95
marks[1]	93
marks[2]	98
marks[3]	92
marks[4]	97

CST8116 Review

This course builds on CST8116, please review this material so you don’t fall behind.

- Questions?
- Comments?
- Feedback?

Multidimensional Arrays

What if you need to express your information in more than one dimension?

Example:

- Desk row/seat number
- Building/floor/room
- Year/month/day/hour/minutes

Seating Arrangement

	Column 1	Column 2	Column 3	Column 4	Column 5
Row 1	**John**	**Doe**	**Bill**	**Tom**	**Sam**
Row 2	**Nora**	**Lily**	**Emma**	**Sophia**	**Maya**
Row 3	**Thomas**	**Leo**	**Jacob**	**James**	**Jack**

15 students to be seated in 3 rows. Each row has 5 seats.

Could we use a single dimensional array?

Multidimensional Arrays

Declaration would be

```
maxRows = 3; maxSeats = 5;
String[][] studentName = new String[maxRows][maxSeats];
```

To get/set student name:

```
String name = studentName[row][seat];
studentName[row][seat]= “John”;
```

A method declaration would be:

```
public void printName(String[][] names) {}
```

Multidimensional Arrays

- `names.length` returns the number of rows
- `names[0].length` returns the number of seats
- `names[3][4]` only mean row 3 seat 4
- `name[2][25]` would create a run time error
- **What about the different sized rows?**

Multidimensional Arrays

For example if each row has twice as many seats as the previous.

```
String studentName[][] = new String[5][];
studentName[0] = new String[1];
studentName[1] = new String[2];
studentName[2] = new String[4];
studentName[3] = new String[8];
studentName[4] = new String[16];
```

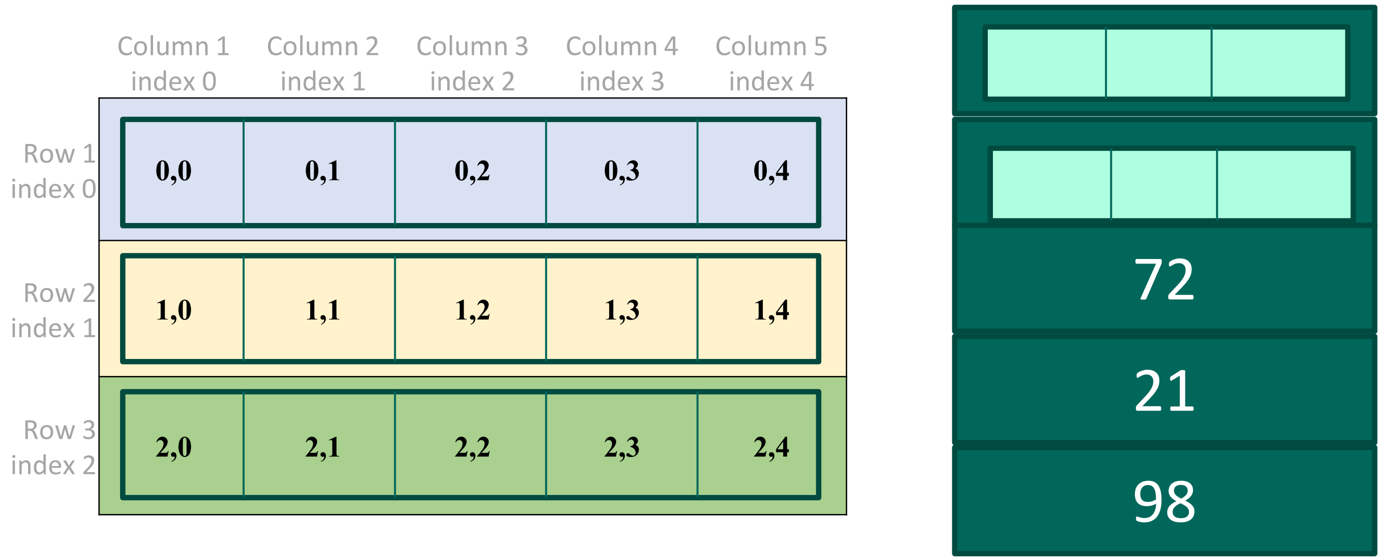
Multidimensional Arrays

- We can even initialize an array

```
String wordLength[][] = {
            {},                                     // one letter numbers
            {},                                     // two letter numbers
            {"one","two","six","ten"},              // three letter numbers
            {"four","five","nine"},                 // four letter numbers
            {"three","seven","eight"},              // five letter numbers
};
```

2-dimensional Arrays

	Column 1 index 0	Column 2 index 1	Column 3 index 2	Column 4 index 3	Column 5 index 4
Row 1 index 0	0,0	0,1	0,2	0,3	0,4
Row 2 index 1	1,0	1,1	1,2	1,3	1,4
Row 3 index 2	2,0	2,1	2,2	2,3	2,4



Two-dimensional array: One-dimensional array with each element as a one-dimensional array

3-dimensional Arrays

- One-dimensional array with two-dimensional arrays
- Example: Multi-level Parking lot
  - Parking lot with different levels, each level with rows and columns of parking spaces
  - Car parked on 5th spot of 3rd row of level 4 ➔ 4th level, 3rd row, 5th position
  - carpark[4][3][5]

Multidimensional Arrays

- Questions
- Comments
- Feedback

In-class exercise (solution will be provided next week)

- Write a class named Numbers with the following:
  - `public void generateNumberTable()`
  - `public void pascalTriangle(int size)`
  - https://en.wikipedia.org/wiki/Pascal%27s_triangle
  - `public void printTable()`
- Write a class named NumbersTest with the following:
  - "`main`" to call the three methods

In-class exercise

```
public static void main(String[] args) {
	System.out.println("\n\nTable of Integers");
	Numbers n1 = new Numbers(5,10);
	n1.generateNumberTable();
	n1.printTable();

	System.out.println("\n\nPascal Triangle");
	Numbers n2 = new Numbers();
	n2.pascalTriangle(5);
	n2.printTable();
}
```

Output:

```
Table of Numbers
   1   2   3   4   5   6   7   8   9  10
  11  12  13  14  15  16  17  18  19  20
  21  22  23  24  25  26  27  28  29  30
  31  32  33  34  35  36  37  38  39  40
  41  42  43  44  45  46  47  48  49  50


Pascal Triangle
   1
   1   1
   1   2   1
   1   3   3   1
   1   4   6   4   1
```