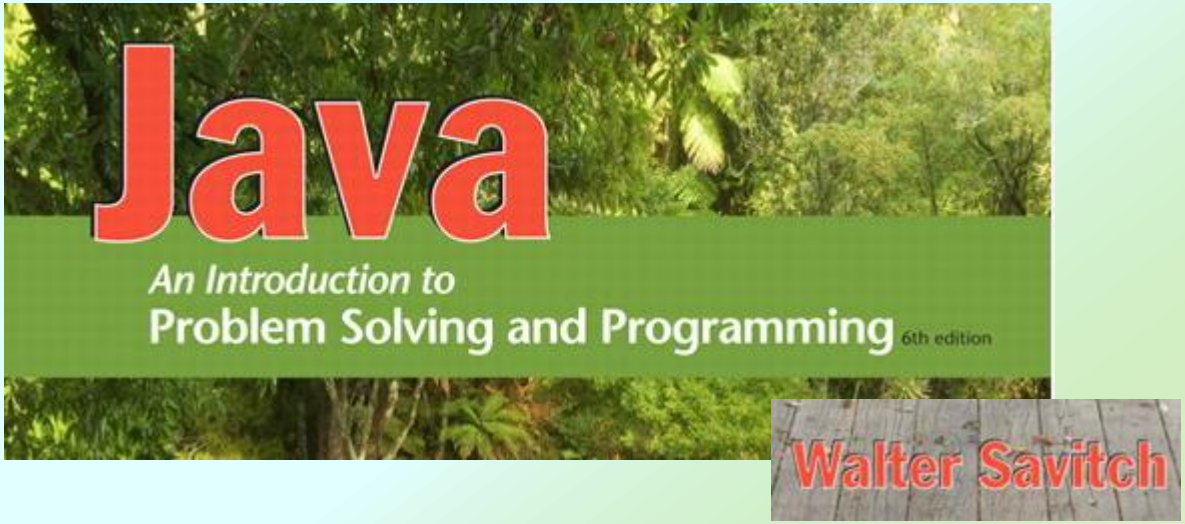

EECS168 Exam 3 Review

*JAVA: An Introduction to Problem Solving & Programming, 6*th Ed. By Walter Savitch
ISBN 0132162709

Exam 3

- Time: 2pm-2:50pm Monday Nov 5
- Closed book, closed notes.
- Calculators or other electronic devices are not permitted or required.

- If you are unable to attend an exam for any reason, no make-up exam will be given. That exam will be dropped from your grade. If a second exam is missed, a make-up exam will only be granted under extenuating circumstances, with prior permission from the instructor.

Exam 3

- Exam 3 covers:
  - **Mostly** Chapter 5 and 6
  - Classes and objects
  - No graphics
  - No Android
  - No Arrays
  - No Javadoc

Exam 3

- How to prepare?
  - Notes, practice questions, homeworks, labs, the textbook!
  - Classes and Objects

Exam 3

- Questions?
  - T/F with justification,
  - Multiple choice, short answers
  - Read code, predict output
  - Debug code: identify syntax errors & logic bugs, fix them
  - **Write code**
  - Other…

Class and Method Definitions

- Java program consists of objects
  - **Objects** of **class** types
  - Objects that interact with one another
- Each **`Java`** class definition usually in a file by itself
  - File name begins with name of the class
  - Ends with .**`java`**
- Class can be compiled separately

```java
public class Dog
{
    public String name;
    public String breed;
    public int age;

    public void writeOutput()
    {
        System.out.println("Name: " + name);
        System.out.println("Breed: " + breed);
        System.out.println("Age in calendar years: " + age);
        System.out.println("Age in human years: " + getAgeInHumanYears());
        System.out.println();
    }

    public int getAgeInHumanYears()
    {
        int humanYears = 0;
        if (age <= 2)
        {
            humanYears = age * 11;
        }
        else
        {
            humanYears = 22 + ((age-2) * 5);
        }
        return humanYears;
    }
}
```

Class and Instance Variables

- Class has
  - Data: member variables (instance variables)
  - Operations (behaviors): member methods
    - Method definitions appear inside class definition
    - Can be used only with objects of that class
- Each instance of this type has its own copies of the data items
- **`public`** and **`private`**

Defining Methods

- Modifiers
  - public or private
  - Return type or void
    - Return a single item (must declare type), last statement: return
      - What if I have statements after (unconditioned) return?
    - Perform some action: no return value, e.g., write output
  - static or non-static
- Parameters
- Body enclosed in braces { }

Methods

- When you use a method you "invoke" or "call" it
- Two kinds of Java methods
  - Return a single item
    - Could be used anywhere a value can be used
  - Perform some other action – a **`void`** method
    - E.g. write output
- The method **`main`** is a **`void`** method
  - Invoked by the system
  - Not by the application program

Methods

- Local variables
  - Variables declared inside a method are called local variables
  - May be used only inside the method
- Blocks
  - Compound statements: enclosed in braces { }
  - The scope of variables declared in the block is from its declaration to the end of the block
  - Variable declared outside the block usable both outside and inside the block

Methods

- Parameters
  - Formal parameter: in the declaration
  - Actual parameter: from the caller
- Parameters of primitive type
  - Names of formal parameters are local to the method
  - When method invoked
    - Each parameter initialized to value in corresponding actual parameter
    - Primitive actual parameter cannot be altered by invocation of the method
  - Type conversion

Information Hiding

- Programmer using a class method need not know details of implementation
  - Only needs to know *what* the method does
- Information hiding:
  - Designing a method so it can be used without knowing details
- Also referred to as *abstraction*
- Method design should separate *what* from *how*
- Encapsulation
- Accessor and Mutator Methods

Encapsulation

- Declare all instance variables in the class as private.
- Provide public accessor methods to retrieve data
- Provide public methods manipulating data
  - Including public mutator methods.
- Make any helping methods private.

Constructors

- A special method called when instances are created with **`new`**
  - Reserve memory for member variables
  - Initialize values of member variables
- Can have parameters
  - To specify initial values if desired
- May have multiple definitions
  - Each with different numbers or types of parameters
- Default constructor

Static Variables

- Static variables are shared by all objects of a class
  - *class variables*: only one instance of the variable exists (note: this is not variables of class type)
  - It can be accessed by all instances of the class
  - Variables declared **`static final`** are considered constants

Static Methods

- Some methods may have no relation to any type of object
- Static method declared in a class
  - Can be invoked without using an object
  - Instead use the class name
  - ClassName.MethodName(parameter1, …);

Overloading Basics

- When two or more methods have same name within the same class
- Java distinguishes the methods by number and types of parameters
  - If it cannot match a call with a definition, it attempts to do type conversions
- A method's name and number and type of parameters is called the *signature*

Variables of a Class Type

- Variables are implemented as memory locations
- Data of *primitive type* stored in the memory location assigned to the variable
- Variable of *class type* (*reference*) contains *memory address of object.*
  - Object itself stored elsewhere in memory
- Compare objects
- Assignment operators for variables of class type
- Parameters of class type

Example 1: Number Class

- Design a class for natural numbers
  - Similar to the wrapper class `Integer`
  - Member variable
    - the number (as int)
  - Member functions
    - the constructor
    - A Boolean method to determine if the instance variable is prime
    - A static method that takes in an int, and determines if it is prime

```java
public class MyNumber {
    private int n;

    public MyNumber(int theN) {
        n = theN;
    }

    public int getN() {
        return n;
    }

    public Boolean isPrime() {
        if (n == 2)
            return true;
        else if ((n % 2) == 0)
            return false;
        for (int i = 3; i <= Math.sqrt(n); i = i + 2) {
            if (n % i == 0)
                return false;
        }
        return true;
    }
```

```java
    public static Boolean checkPrime(int x) {
        MyNumber my = new MyNumber(x);
        return my.isPrime();
    }

    public static void main(String args[]) {
        for (int i = 1; i <= 30; i++) {
            MyNumber myN = new MyNumber(i);
            System.out.println(myN.getN() + " is prime? " + myN.isPrime());
        }
        for (int i = 1; i <= 30; i++) {
            System.out.println(i + " is prime? " + MyNumber.checkPrime(i));
        }

    }

}
```

*JAVA: An Introduction to Problem Solving & Programming, 6*th Ed. By Walter Savitch
ISBN 0132162709

Example 2: TopTwo

- Design a class to trace the two largest numbers
  - Member variables
    - Two numbers (as int)
  - Member functions
    - put(int x): compare x with the two existing numbers, keep the largest two.
    - getLargest() return the larger of the two numbers
    - getSecond() return the smaller of the two numbers

```java
public class TopTwo {
    private int theLargest;
    private int theSecond;

    public TopTwo() {
        this(0, 0);
    }

    public TopTwo(int a) {
        this(0, a);
    }

    public TopTwo(int a, int b) {
        theLargest = 0;
        theSecond = 0;
        put(a);
        put(b);
    }

    public void put(int theNew) {
        if (theNew > theLargest) {
            theSecond = theLargest;
            theLargest = theNew;
        } else if (theNew > theSecond) {
            theSecond = theNew;
        }
    }
```

```java
    public int getLargest() {
        return theLargest;
    }

    public int getSecond() {
        return theSecond;
    }

    public static void main(String[] args) {
        TopTwo t = new TopTwo(9, 8);
        t.put(25);
        t.put(22);
        t.put(7);
        t.put(31);
        System.out.println("Largest: " + t.getLargest());
        System.out.println("Second: " + t.getSecond());

    }
}
```