# Java Review

Topics this week:

- Week 1 - Elements of the Java Platform
- Week 2 - Basic Elements of Java Programming
- Week 3 - Java Arrays, Objects, Methods
- Week 4 - Java's Object Orientation, I/O
- Week 5 - Composition, GUIs, and Applets
- Final Exam



---

# Week 1 - Elements of the Java Platform

## Topics

- Programming
- Elements of the Java Platform
- Creating a Simple Java application
- Installing and running the JDK

## Programming

### **Question:** *What are Programs?*

### **Question:** *What is Programming?*

## Why Java?

- Platform independent
- Simple to learn
- Built-in security
- Designed for internet

## Keys to Java Programming

- In Java *everything* is an *Object*.
- Java programs consist of one or more objects.

**Question:** *What's an "object"?*

- One object is the starting point of the program.
- It is this object's *main* method that starts the application going.

  All Java *applications* have at least one method named "*main*"

## Java tools

- Compiling Java source - Using the *javac* compiler
- Running a Java application Using the *java* interpreter which executes Java bytecodes in a Java Virtual Machine (JVM)

# Week 2 - Basic Elements of Java Programming

## Topics

- Java Platform
- Reserved Words - Primitive data types
- String and System objects
- Expressions, Arithmetic Operators, Assignment Operators
- Reserved Words - Control flow
- Objects and Methods

## Java Platform

- Language
- JDK - Application Programming Interface (API)
- Java Virtual Machine

| Java Program (application, applet, servlet) |
|---|
| Java Application Programming Interface (standard packages) |
| Java Virtual Machine |
| Hardware |

## The Language

The syntax and constructs for writing Java code

## Keywords

The vocabulary of Java

## Primitive Data Types

- `boolean`
- `byte`
- `char`
- `double`
- `float`
- `int`
- `long`
- `short`

## Control flow - Branching

### Branching

-
```
if (expression that evaluates to a boolean value) {
  ...
}
```
-
```
if (expression that evaluates to a boolean value) {
  ...
}
else {
  ...
}
```
-
```
switch (expression that results in an integer value) {
case value:
  ...
  break;
```

```
...
default:
  ...
  break;
}
```

# Classes

## Question: *What is a class?*

**A user or API defined data type. A blueprint for objects.**

# Objects

## Question: *What is an object?*

**an instance (occurrence, instantiation) of a user or API defined data type.**

# Constructors

## Question: *Purpose? What distinguishes them?*

# Week 3 - Java Arrays, Objects, Methods

## Topics

- Control Flow - Loops
- References to and Creating Objects
- Arrays
- Objects - Instances of classes
- Java Methods
- Introduction to Inheritance

## Control Flow

### Looping

- ```
  for (initialization ; (expression that evaluates to boolean ; increment) {
     ...
   }
  ```
- ```
  while (expression that evaluates to boolean) {
     ...
  }
  ```
- ```
  do {
    ...
  } while (expression that evaluates to boolean);
  ```
  ```

# Objects

- An object is a software bundle of variables and related methods.
- Instances of objects are created using the **new** keyword.
- Results in a *reference* to an instance of an object.

# Arrays

Groups of similar elements, both primitives and object references.

```
int intArray[] = new int[4]; // elements initially set to 0

CreditCard cards[] = new CreditCard[MAXCARDS];
                  // elements initially set to null
```

# Methods

- Instance Methods
  - methods that are called by referencing an instance of a class

```
ClassName someObject = new ClassName();
...
someObject.someInstanceMethod( ... );
```

- Class Methods
  - Associated with an entire *class*, not a single *instance* of a class.
  - Identifiable because they are declared **static**
  - Class methods are always callable; there doesn't need to be an instance of the class for you to call class methods on, you just need access to the class and the desired method.

```
ClassName.classMethod( ... );
```

**Question:** *Which one is "*better*"? Why?*

---

# Week 4 - Java's Object Orientation, I/O

## Topics

- Methods
- Overloading
- Garbage Collection
- Accessability
- Encapsulation
- Inheritance
- Input/Output

## Methods and fields

- Class methods vs. Instance Methods
- Class variables vs. Instance variables vs Local variables

### Overloading

**Question:** *What is it? Why is it useful?*

### Garbage Collection

**Question:** *What does it mean?*

### Accessability modifiers

**Question:** *Purpose? What are the Java accessability modifier keywords?*

# Object Oriented Programming

## Encapsulation

- An instance of an class (an object) should contain just what is needed to model a real world thing to the necessary detail to accomplish a problem task.
- Implementation details should not be accessible.

## Inheritance

- An object type can provide additional details (specialization) and capabilities than are provided by a more generic type.

  **Question:** *What is the Java keyword that signals you are using inheritance?*

- Creating **abstract** and generic classes and then sub classes that inherit from them can be useful to reduce code duplication.

# Input/Output

Using **java.io** classes:

- Character Streams - Readers and Writers
- Examples of reading file contents
- Reading Strings
- Converting characters in Strings into primitive data

# Exceptions

**Question:** *What are they? How do you deal with them?*

# File I/O, handling exceptions, interpreting contents

# Week 5 - Composition, GUIs, and Applets

## Topics

- Abstract Windowing Toolkit (AWT)
- Interfaces
- Event Driven Programming
- Graphical User Interfaces
- Applets

## Abstract Windowing Toolkit

- Basic graphical user interface components:
  - Menus
  - Buttons
  - Frames
  - Canvases
- Succeeded by *Swing*

## Interfaces vs. Inheritance

- Inheritance - **extends** some other class' capabilities
- Interfaces - are like a mini-contract for methods a class *must* implement

## Event Driven Programming

Typical of most modern programming applications:

```
while (true) {
  event = waitForSomethingInterestingToHappen();
  callThingThatRegisteredInterest( event );
}
```

Java Graphical User Interfaces rely on classes that `implements` *`interface`*

## Final Exam

The final exam for previous CS 161 Java classes consisted of approximately 50 questions on various aspects of programming and Java. I intend to make the final for this session similar, but the questions for this session will be derived more closely from the material presented in the textbook and class.

Things to note:

- My exams are generally true/false, short answer, multiple choice, or you are asked to write some code fragments.
- Code fragments should be just that: I'm looking for your understanding of simple coding constructs, not for a complete program. I'm not too particular that the code you write would pass the compiler without error, though I do expect your code fragments to be reasonably correct.
- Read the questions carefully: case can be significant. Read the whole question and all multiple choice answers. There is one response that I think is most correct.
- If you think there is more than one right answer (or no correct answer) then write a note explaining your thinking.
- Hint: Do the whole exam before giving up on a question: sometimes the answer will be found in later or previous questions.

- When in doubt, guess. I don’t penalize for wrong answers but I give no credit for blanks. Be sure every question has a response.
- I create several versions of the final exam, all with the same questions but in different random order. *Please spend your time preparing for the final rather than trying to figure out ways to out-smart me*.

Here is a link to a copy of a set of CS161 exam questions and answers from last Spring.

http://cs.gmu.edu/~jdoughty/cs161/Exam5.html

## Material Covered by Exam

- Textbook
  - Chapter 1 - Introduction and a Taste of Java
  - Chapter 2 - Primitive Types and Strings
  - Chapter 3 - Flow of Control
  - Chapter 4 - Classes, Object, and Methods
  - Chapter 5 - Programming with Classes and Methods
  - Chapter 6 - Inheritance - through the section "Constructors in Derived Classes" that ends on page 301 (you may skip the remainder of Chapter 6)
  - Chapter 7 - Event Driven Programming Using the AWT
  - Chapter 8.1 - Basic Exception Handling
  - Chapter 9 - Streams and File I/O - Sections 9.1 and 9.2

- Class Notes
  - HTML versions: cs16101.zip cs16102.zip cs16103.zip cs16104.zip cs16105.zip
  - PDF versions java01.pdf java02.pdf java03.pdf java04.pdf java05.pdf