Final Exam Review

CSE 114 INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING

Announcements

Final Exam: Monday, December 7 from 12:30-3 pm

Today: Review for Final

Topics

Java basic features, syntax, operations, flow constructs

Java arrays

Java classes

Object-oriented programming features

- Interfaces
  Encapsulation
- Inheritance
- Polymorphism

Java API [Java libraries]

Java Basic Features

Data types:

- int – values from around -2000000000 to 2000000000
- double – values with fractions
- Boolean – values: true/false
- Char – Single character
- String [class] – Series of characters

Math operations

- + - * / %

Compound operators (+=, *=, etc)

Math class (Math.abs(), Math.sin(), Math.cos(), etc)

Java Basic Features

Relational Operators:

- <, >, <=, >=, ==, !=

Logical Operators

- &&, ||

Java Arrays

Collection of objects of same type

- Order is significant
- Indexed by an integer value from 0 to n-1 (size of array is n)

Arrays can have more than 1 ‘dimension’ [no limit on number of dimensions]

Java Arrays

Declaration (examples)

- int[] newArray = new int[20];   // create array of 20 integers.
- int newArray[] = new int[20];   // same

- int[] newArray;   // same but in two separate lines
- newArray = new int[20];

Use

- int y = newArray[2];   // copy 3rd element into y (remember arrays start indices from 0)
- int a = y * newArray[10] + 15;   // Use an array element in a mathematical expression
- newArray[0] = y - a;   // set an array value

Java Arrays - Multidimension

```
Int[][] anotherArray = new int[20][10];
```

ArrayLists

More convenient class than using arrays

Can dynamically add / delete items and resize

Contains additional methods

Java Classes

Contain:

- Fields (members)
- Methods
- Constructors – Special method to initialize an object
- A way to encapsulate the state and behavior of an entity

Methods

Models behavior of objects

By invoking methods in an object, you are accessing (viewing/modifying) the state of the object

Accessor methods (getters/readers)

Mutator methods (setters/writers)

Parameter passing in Java

- passing primitive values by value
- passing object references by value

Names

Class names

Method names

Variable names

- Fields (members) in a class
  - Static variables (class variables, static fields)
  - Non-static variables (instance variables, dynamic fields)
- Local variables in a method
  - Parameters and local variables are treated the same

Visibility

How do we control the visibility of the state information in an object?

How does this relate to the concept of information hiding?

How does this relate to the concept of encapsulation?

How does the concept of a class relate to the concept of encapsulation?

private

protected

public

Scope and Lifetime

Scope and lifetime of a static field

Scope and lifetime of a non-static (dynamic) field

Scope and lifetime of a local variable

Scope and lifetime of a parameter

- What are the differences between a local variable and a parameter? Not much!

Primitive and Non-primitive Types

Primitive – int, double, boolean, etc

Non-Primitive – classes (Point, Student, Account, etc)

String

Not primitive data type

Variety of operations and methods:

- +
- substring
- charAt
- toLowerCase
- length
- split

What is an expression in a programming language like Java?

An expression produces a value when evaluated

A value is stored in a location in memory

Computation is really storing values in memory locations in some meaningful ways (storing and updating as computation continues)

Assignment statement (the '=' operation) is how you store/update a value in a memory location

Operator Precedence

What is the value of the following expression?

3 + 4 * x - y / 4 + 4

Java Flow Control

Selection – if/else, switch

Loops – for, for(each), while

Exceptions – try/catch

Conditionals in Java

if...

if...else...

Looping

while
- for
- for (‘for-each’)
- Using recursion to loop/iterate

Loops and Arrays

Use of while or for or recursion to iterate over elements in an array (1D or 2D array) or an ArrayList is a natural thing to do!

Data Structures

Objects containing fields of various types including

- object types
- Arrays containing data/objects of a certain type
- linear arrays
- 2D arrays
- arrays are objects!
- array of primitive type values
- array of objects

Is Everything in Java an Object?

No, primitive data type values are not objects, e.g., int, double, boolean, char, etc.

- However each of those primitive data type has an object version, e.g., Integer for int, Double for double, Character for char, etc.

So, the only data type values that are not objects are those primitive data type values, so the statement above is almost true.

Exception Handling

try . . . catch . . .

I/O

Standard I/O

- Keyboard input
  - System.in
  - java.util.Scanner
- Console/Screen output
  - System.out.print
  - System.out.println
- File I/O
  - Open a file
  - Read from or write to it
  - Close it

Memory models [What/Where are they stored?]

Primitive data types in an object

Primitive data types in a method as a parameter or a local variable

Object reference in any context: a variable of any object type is really just an object reference of 32 bits and the actual object is sitting in another location in memory (in the heap) referenced by those 32 bits of object reference

Fields in an object (static vs. non-static)

Arrays (Arrays are objects too)

Method Overloading

```
foo(int x);

foo(boolean x);

foo(int x, double y);
```

Constructors are often overloaded too

Method calls (static/non-static)

The dot (.) notation

Internal calls within a class do not need the '.' notation

External calls from outside a class use the '.' notation

A dot used with a dynamic object

A dot used with a static object

- A class name followed by a dot followed by a method name with actual arguments
- That class name as an object is a meta-object (an instance of the class named Class).
  - (You are not responsible for understanding this concept for the final exam.)

this/super

this.

this(. . .) [to call another constructor in a class]

super.

super(. . .) [to call a constructor of the superclass]

Concepts in OO Design

Object composition vs. inheritance

Composition models a 'has-a' relationship between objects

Inheritance models an 'is-a' relationship between objects

Software Development Methods

Incremental development

Test-first development

Debugging a Java program

Using IntelliJ

- Break points
- Stepping

Using System.out.println

(You will not be tested on these in the final exam)

Bits and bytes

A byte is 8 bits. Computer stores everything as bits.

We covered base 2 numbers and base 10 numbers

(You will not be tested on these in the final exam)

Character Codes

ASCII code (7~8 bits)

Unicode

(You will not be tested on these in the final exam)

Libraries in programming languages

packages in Java

import ...

java.lang.*

java.util.*

java.io.*

. . .

Constants in Java

Use of final

- public static **final** double PI = 3.141592653589793;

A class may be declared to be final also

Equality

==

vs.

equals

Parsing with split() in String

Use of regular expressions

Example: [!, .;:]+

Interfaces

`public interface <name> { }` // Contains ONLY constants and method headers

Interfaces are ‘implemented’

Design of a type hierarchy (using interfaces and classes)

Subtyping

Examples:

- **Comparable** with **compareTo**
- **Shape** with **translate**, **scale**, **boundingBox**, etc.

Abstract Classes

public abstract class SomeClass { … }   // Partial implementation of a class

Classes with 1 or more methods unimplemented (abstract)

Cannot be instantiated but can be extended to create a concrete class which can

- Concrete class must implement all the abstract methods

Inheritance

public class ChildClass extends ParentClass { }

- Child acquires all members/methods in parent
- Can add new methods or override parent’s methods

The Object class has methods:

- toString
- equals

Models an 'is-a' relationship with inheritance

- cf. object composition ('has-a' relationship)

Inheritance example: Person, Employee, Manager

Can use **protected** visibility control

Inheritance (cont)

Overriding a method definition in a subclass

Polymorphism

- Polymorphic object references
- Method name polymorphism

Dynamic binding

- Allows at run-time to run the appropriate method in the actual type of an object, not the declared type
- Multiple names for the same thing: Dynamic method name lookup, dynamic method dispatching, late binding, dynamic binding, run time binding
- Different compared to: static binding, early-binding, compile-time binding

Types

Declared type vs. actual type

- static type (aka, declared type)
  - Used by compiler as it typechecks a piece of code at compile time
- dynamic type (aka, actual type)
  - Used by method dispatching to type check a value at run time

Type casting and type downcasting

What do we gain by using inheritance?

Recursion

Think recursively!

Recursion as a control mechanism - can use it for looping

Learn different patterns

Need more exercises on recursion? See Exercises in Chapter 8 of Downey

Sorting

Selection sort : $O(n^2)$

Insertion sort : $O(n^2)$

Sorting an array of primitive type data

Sorting an array of objects

- using compareTo in Comparable

You will not be responsible for the big-O notation in the final exam

Searching

Linear search: O(n)

Binary search: O($\log_2$ n)
- both iteratively and recursively
- using primitive data or objects

Searching an item in an array of objects
- Using equals
- using compareTo in Comparable

Performance concerns in search
- How can we improve the searching time?
  - If we sort the data, we can use binary search
- More on this topic next semester

Questions?

Thanks for your attention and hard work this semester!