CS102

Introduction to data structures, algorithms, and object-oriented programming

May 9, 2016
• The standard Java library contains many packages. The only one that is imported automatically is java.lang. imports can be used for others.

• Primitive types: int (short, long, byte), double (float), char, boolean.

• Reference types or Class types
  static modifier for class variables and methods are called with the class name to the left of the period
  – . (period) operator goes inside an object or a class to invoke a particular method or to access a particular variable. Period is preceded by the class name if the member is static and by the object name if the member is non-static.
  – instance variables and instance methods can only be called on an object

• Java classes are either a library of static methods or a template for objects (or really any combination of the two).
• Java naming conventions.
• Java keywords.
  – boolean break
  – catch char class
  – double
  – else extends
  – final for
  – if implements import instanceof int interface
  – new
  – package private public
  – return
  – static super switch
  – this throw throws try
  – void
  – while
• All variables must be declared as a particular type before they can be used in an expression.
• Local variables are declared inside a method or in the method parameter list and cannot be modified as public, static, or private.
• Global variables can be instance or static variables and are declared inside a class, but not inside a method.

Java classes contain these categories of members directly inside the class braces {}:
  • instance or class variables
  • class constants
  • instance or class methods
  • inner classes
Every class that can run on its own must have a main method, with signature:

```java
public static void main(String[] args)
```

Unless you specifically intend a program to be used as a template for objects, you can in most situations declare all members to be static. However, if you are overriding methods in an interface, abstract class, or superclass, you must use the method signature as it exists in the interface, abstract class, or superclass.

Operators include: + - * / = <= >= != && || ! instanceof .

When comparing two primitive type variables, use the `==` method, but when comparing object (aka reference) types, use the `.equals()` method.
Every template-type class you write should have the following methods overriding the methods provided in the Object superclass:

```java
public String toString():
    returns a String representation of this object
public boolean equals(Object other):
    returns true if this and the other object are equal
```

There are many more methods in the Object class and you can overwrite those as needed.

Printing to the standard output using classes from java.lang:

```java
System.out.print, System.out.println, or System.out.printf
```

Printing to pop-up boxes uses classes from javax.swing:

```java
JOptionPane.showMessageDialog
```

Reading from pop-up boxes:

```java
JOptionPane.showInputDialog
```

Reading from Scanner class
Looping constructs in Java: **while, do while, for, recursion**

Nested looping: for loop inside a for loop. While inside a while.

Loop and a half uses **break** inside while (true).

Branching statements: **if, else if, else. If, else if, If alone, ?: switch**

**Array** declaration, instantiation, and use.

**2-dimensional arrays**

Generating random numbers.

Type-casting
Looping constructs in Java: while, do while, for, recursion

Nested looping: for loop inside a for loop. While inside a while.

Loop and a half uses break inside while (true).

Branching statements: if, else if, else. If, else if, If alone, ?: switch

Array declaration, instantiation, and use.

2-dimensional arrays

Generating random numbers.

Type-casting: Implicit and explicit
Java memory organization

ArrayList<E>

for each loop over collections

Passing information in and out of methods

Static vs non-static methods

Syntax used for calling methods and writing method definitions.

Overriding methods

Returning values from methods

JUnit tester classes (probably not covered on exam).
Data encapsulation

Data in Java is passed by value into methods. This value has a different meaning for primitive and reference types.

Arrays of objects.

Inheritance,
  UML diagrams
  Overriding methods
  using the extends keyword
  abstract classes
  super keyword

2 meanings of interface
Polymorphism
method stubs
Interfaces

Structural recursion using interfaces

Singly-linked lists

Generic types

Stacks

Exceptions
  try...catch
  throws
  throw
  writing Exception classes
Accessing static vs non-static members in classes.

Pass-by-value of primitive and reference types.

Arrays of objects

Parameterized (generic) classes

Stack ADT
  throw and throws keywords
  exception handling
  exception hierarchy

Reading from and writing to files

Deques and doubly-linked lists
Iterable interface

Dynamic binding and polymorphism

Reading from the command line and file I/O
   StringTokenizer

Writing Graphical User Interfaces (GUIs)
   Adding ActionListeners
   Basic GUI components
   Drawing in a graphics application
   Animating applications
   Inner classes and anonymous inner classes
   Extending JFrame and JPanel
   Adding MouseListeners
   KeyListeners and FocusListener
Final Exam Reminders

• Read Chapters 1-10, and 13 of our on-line textbook.

• Final exam: Friday, May 20th from 5-7 pm, SP 309

You will be allowed to bring your notes to the exam.