**Classes and Objects**

* Objects and classes and the relationship between them
* Understand how to declare an object reference variable and then create the object with new, along with how to invoke the chosen constructor
* Understand how objects are passed in and out of methods
* Constructors:
  + Know what a constructor is for
  + Know how to identify a constructor in a class definition (i.e. the constructor has the same name as the class and no return type)
  + Know how to invoke a constructor (when building an object with new
* The modifiers public and private
* Accessor and mutator methods
* The special toString() method
* The static modifier, and the difference between static variables and methods vs. instance variables and methods, as well as which variables can be accessed from which methods
* The keyword this
* Class scope vs. local scope (like in a method)
* Arrays of objects (array of reference variables, each can attach to an object)
* Object as method parameter (pass by reference)

**Inheritance and Polymorphism**

* Superclasses and subclasses, and deriving a class with the keyword extends
* The use of super to invoke parent constructors and methods
* Understand overriding methods in derived classes
* Every class is derived in some way from class Object. Understand these 3 special methods inherited from Object by every class:
  + equals()
  + toString()
  + clone()
* Understand the modifiers protected and final, and what they do when applied to class data and methods
* Abstract classes and their relationship with abstract methods
* Understand polymorphism and dynamic binding
  + How base class variables can be attached to derived class objects
  + How base class methods can be overridden in derived classes
  + How dynamic binding allows method calls (through base class variables) to still invoke child versions of overridden methods
  + When casting is necessary between parent and child, and when it is not
* ~~Understand interfaces, how to declare one and how to extend one~~
* ~~Know how a class declares that it will implement an interface~~
* ~~Understand relationship of data and methods with interfaces~~
* ~~Understand the basic interfaces discussed in examples (Comparable and Clonable)~~

**Programming hints**

* You may assume that java.util.Scanner is available to be used
* Be able to write methods that do simple algorithmic tasks (like in the homeworks) and use parameters and returns appropriately
* Given an array be able to do a variety of array processing tasks -- a few examples are: compute or find and print out the sum of the elements, the product of the elements, the average of the elements, the maximum element, the minimum element, etc.
* Be able to write a method that modifies the array sent to it. A method that return the array but with the elements sorted would be an example
* Be able to write a method that modifies an object sent to it. Also be able to write a method that returns a new object (based perhaps on some relation to a parameter object or a calling object).
* Be able to write code statements that involve relationships between base and derived types

**class** Student {

// instance variable (non-static)

**private** **int** testGrade;

// static variable (class variable)

**private** **static** **int** *numStudents* = 0;

// class constant

**private** **final** **static** **int** *pointsPossible* = 100;

**public** Student() {

testGrade = 0;

}

**public** **void** setGrade(**int** gr) {

testGrade = gr;

}

**public** **int** getGrade() {

**return** testGrade;

}

**public** **static** **void** incrementNumStudents() {

*numStudents*++;

}

**public** **static** **int** getNumStudents() {

**return** *numStudents*;

}

}

In this sample code:

* **testGrade is an intance variable**. Each object of type Student will have its own copy of testGrade
* **numStudents is a class varaible (static)**. There is only one variable shared by the whole class. **The variable's value can be changed, but changes are seen by all objects**
* **pointsPossible is a class constant**. There is only one variable (because of static), and its value cannot be changed
* **setGrade and getGrade are instance methods**. **They must be called through individual objects**
* **incrementNumStudents and getNumeStudents are static methods**. They cannot access instance variables of the class, but they can be called through the class name, regardless of whether any objects have been created