Classes & Object: Vocabulary & Concepts Summary*

Class: A program entity that represents either:
   1. A program / module, or
   2. A template for a new type of objects.

Abstraction: A distancing between ideas and details. We can use Objects without knowing how they work, just like you don’t need to know how an iPod’s circuitry works just how to use it’s control panel (methods) and the settings (fields).

Client Program (Class): A program that uses Objects
Object (Class): An entity that combines state and behavior.
   1. State: Fields (i.e. int, double, array, other objects)
   2. Behavior: Methods
      a. mutator: A method that modifies an object’s state. Examples: setLocation, translate. Typically has a void return type
      b. accessor: A method that lets clients examine object state. Examples: distance, distanceFromOrigin. Typically has a non-void return type

Instance method (or Object method): Exists inside each object of a class and gives behavior to each object.
   
   public type name(parameters) {
   statements;
   }

Implicit parameter: The object on which an instance method is called.
   p1.draw(g);
   the object referred to by p1 is the implicit parameter, the method will act on its fields

The instance method can refer to that Object's fields

Constructor (method): Initializes the state of new objects. (where type is the Object’s name)
   public type(parameters) {
   statements;
   }

A class can have multiple constructors. But each one must accept a unique set of parameters. BTW, this is also true of methods in Object Classes and Client Programs.

Array of Objects: you can create an array of any kind of objects, but the elements of an array of objects are initialized to null.
   null: A value that does not refer to any object (yet).

dereference: To access field data or methods of an object with the dot notation:
   such as a field: p1.x or method: s.length()
   –It is illegal to dereference null (causes an exception).
   –null is not any object, so it has no methods or data.

*Assembled from Supplement Lesson slides of: Building Java Programs, Chapter 8 by Stuart Reges and Marty Stepp (http://www.buildingjavaprograms.com/ )
Array of Objects requires two-phase initialization:
1) initialize the array itself (each element is initially null)
2) initialize each element of the array to be a new object

```java
String[] words = new String[4]; // phase 1
for (int i = 0; i < words.length; i++) {
    coords[i] = "word" + i; // phase 2
}
```

Arrays as Object Fields: when the length of an Object’s array field is not known till constructed:
• First just declare it like you would any variable with no size:
  ```java
  // declare our fields (data)
  private int numClasses;
  private int[] scores; // Note: No Size
  ```
• And then “Instantiate” it in the constructor method using the length available parameter.
  ```java
  // Constructor for Object GradeBook
  public GradeBook (int numClasses) {
      this.numClasses = numClasses;
      // We now "Instantiate" the array to its size
      scores = new int [numClasses];
  }
  ```

toString method - tells Java how to convert an object into a String
```java
public String toString() {
    code that returns a String representing this object;
}
```
Every class has a toString, even if it isn’t in your code. Default is: class’s name @ object’s memory address (base 16)
Point@9e8c34

Encapsulation: Hiding implementation details from clients using Private Fields.
• Encapsulation forces abstraction.
• separates external view (behavior) from internal view (state)
• protects the integrity of an object’s data

Private Field: A field that cannot be accessed from outside the class
```java
private type name;
```

Shadowing: 2 variables with same name in same scope. Normally illegal, except when one variable is a field.
```java
public class Point {
    private int x;
    private int y;

    public void setLocation(int x, int y) {
        ...
    }
}
```

this: Refers to the implicit parameter inside your class. (a variable that stores the object on which a method is called). “this” can fix (clarify) variable shadowing:
```java
public void setLocation(int x, int y) {
    this.x = x;
    this.y = y;
}
```

*Assembled from Supplement Lesson slides of: Building Java Programs, Chapter 8 by Stuart Reges and Marty Stepp (http://www.buildingjavaprograms.com/)