Pages 206 ***Java Programming A Comprehensive Introduction***

**Section 1: Define / Answer**

Encapsulation- Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.

Encapsulation is the technique of making the fields in a class private and providing access to the fields via public methods. If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class. For this reason, encapsulation is also referred to as data hiding.

Encapsulation can be described as a protective barrier that prevents the code and data being randomly accessed by other code defined outside the class. Access to the data and code is tightly controlled by an interface.

The main benefit of encapsulation is the ability to modify our implemented code without breaking the code of others who use our code. With this feature Encapsulation gives maintainability, flexibility and extensibility to our code.

default- The "default" access (specified by the absence of a keyword) is also called [**package-private**](http://stackoverflow.com/questions/5416074/java-package-private-terminology).

public- exposes to classes outside the package.

private- private hides from other classes within the package

In very short

* Public are accessible from everywhere.
* Protected are accessible by the classes of the same package and the subclasses residing in any package.
* Default are accessible by the classes of the same package.
* private are accessible within the same class only.

As a rule of thumb:

* **private**: class scope.
* **default** (or *package-private*): package scope.
* **protected**: package scope **+ child** (like package, but we can subclass it from different packages). The protected modifier always keeps the "parent-child" relationship.
* **public**: everywhere.

As a result, if we divide access right into three rights:

* **(D)irect** (invoke from a method inside the same class).
* **(R)eference** (invoke a method using a reference to the class, or via "dot" syntax).
* **(I)nheritance** (via subclassing).

then we have this simple table:

Task 1:

**USE OBJECT ORIENTATED PROGRAM DESIGN TO SOLVE PROBLEM**

Update Assignment #10, Task 1.

Create a Parent SuperClass Student. Containing First Name, Last Name, DOB, Social Security Number.

Create a subclass containing Street Address, and Zip Code

Create a subclass containing Student ID number, Major

The program should execute in way that student objects are created. Then create a menu where the user can print various portions of information about a given student.

Override the method for printing in each class to display the required print information.

Create private modifiers for sensitive materials.

Return redacted versions of social security, DOB.

For example – Social security = XXX-XX-8010

DOB – XX/XX/1980

Attach Snipping Photos Below

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Main Menu: \***

**\* Enter # to run program or Quit \***

**\* 1) Print Student Name \***

**\* 2) Print Student Address \***

**\* 3) Print all Student info \***

**\* 4) Quit \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

