Students who contributed to website:
1. Nour Mikhael (First section)
2. Kevin Aoun (First section)
3. Salim el Elliye (First section)
4. Lea Haidamous (First section)
5. Joy Andraos (Second section)
6. Georgio Nassar (Second section)
7. Riwa Hoteit (Second section)
8. Charbel Albateh (Second section)
9. Yusuf Alhajj (Second section)

- Inheritance: Object oriented concept + Polymorphism: Principle
Encapsulation + Interfaces
Foundation of design patterns (23 by GoF).

What is inheritance?
Objective: create a new class (child, subclass, or derived) based on an existing one (Parent, superclass, or base class).

Parent class <--is a-- Child class
If X is derived from Y, then X is a Y.

Example#1: InheritanceDemo

Book (parent) <-is a-- Dictionary (child)

Inheritance support the idea of "Software reuse". Child class inherits all the methods (except for the constructor) and all the variables that are defined inside the parent class.

In Java, all classes are either directly or indirectly derived from the Object class.
Java does not support multiple inheritance. A class cannot have more than one parent.

Lessons pertaining to first example:
1. Child class inherits all variables, including private one
2. Isolation of concerns:
3. super allows the child class to call the constructor of the parent class.
4. protected (package access) variables can be accessed by name by any class that belongs to the same package.

- Method overriding vs overloading

Can the child class create a method that has the same signature as one defined inside the parent class?

It is the process of redefining an inherited method.

```
Shape (superclass): draw <---- Circle
^
|
Rectangle
```

Lessons pertaining to second example:
1. Based on the data type of the object, I would know which of the three versions of the draw method will be executed. (WRONG) It depends on the type of the object being created.

Principles:
1. We code to an interface ---> Abstraction
foo(List)
2. Single responsibility principle ===> code smell

```
int roll() {
        faceValue = (int) Math.random() * 6 + 1;
        return faceValue;
}
```

```
3. Refactoring ==> get rid of the code smells
4. Open (extension) closed (for modification) class

- toString + equals
obj1.equals(obj2) ==> equals will check whether obj1 and obj2 are aliases.
Example#3: EqualsOverridingDemo
```