# Summary points you should/hopefully acquired through experimentation and code samples

* You can derive a new class from an existing class (Class Inheritance). The new class is called a **subclass, child class,**  or **extended class.** The existing class is called a **superclass, parent class** or **base class**

* A constructor is used to construct an instance of a class. Unlike properties and methods, the constructors of a superclass are not inherited in the subclass. They can be invoked ONLY from the constructors of the subclasses, using the keyword **super**.
* A constructor may invoke an overloaded constructor or its superclass’s constructor. The call must be the first statement in the constructor. If none of them is invoked explicitly, the compiler puts **super( )** as the first statement in the constructor.
* To override a method, the method must be defined in the subclass using the same signature as in its superclass.
* An instance method can be overridden only if it is accessible. Thus a private method cannot be overridden, because it is not accessible outside its own class. If a method defined in a subclass is private in its superclass, the two methods are completely unrelated.
* Like an instance method, a static method can be inherited. However, a static method cannot be overridden. If a static method defined in the superclass is redefined in a subclass, the method defined in the superclass is hidden.
* Every class in Java is descended from the **java.lang.Object** class. If no inheritance is specified when a class is defined, its superclass is **Object**.
* If a method’s parameter type is a superclass (e.g. **Object**), you may pass an object to this method of any of the parameter’s subclasses (e.g. **Circle** or **String).** When an object (i.e. a **Circle** object or a **String** object) is used in the method, the particular implementation of the method of the object that is invoked (e.g. **toSubstring)** us determined dynamically.
* It is always possible to cast an instance of a subclass to a variable of a superclass, because an instance of a subclass Is ***always*** an instance of its superclass. When casting an instance of a superclass to a variable of its subclass, explicit casting must be used to confirm you intention to the compiler with the **(SubclassName)** cast notation.
* You can override an instance method, but you cannot override a field (instance or static) or a static method. If you declare a field or a static method in a subclass with the same name as one in the superclass, the one in the superclass is hidden, but it still exists. You can reference the hidden field or static method using the **super** keyword in the subclass. The hidden field or method can also be accessed via a reference variables of the superclass’s type.
* When invoking an instance method from a reference variables, the actual *class of the object* referenced by the variable decides which implementation of the method is used *at runtim*e. When accessing a field or a static method, t6he *declared type* of the reference variable decides which method is used at *compile time*.
* You can use **obj instanceofAClass** to check whether an object is an instance of a class.
* You can use the  **protected** modifier to prevent the data and methods from being access by nonsubclasses from a different package.
* You can use the **final** modifier to indicate that a class is final and cannot be a parent class.

# Practice

1. Find the problems (errors) in the following classes:

**public class** Circle {

**private double** radius;

**public** Circle(**double**  radius) {

**this.radius = radius;**

**}**

**public double** getRadius( ) {

**return** radius;

**}**

**public double** getArea( ) {

**return** radius \* radius \* Math\_PI;

**}**

**}**

**class B extends** Circle {

**private double** length;

**B(double** radius, **double** length) {

super(radius);

this.length = length;

}

/\*\* Override getArea() \*/

**public double** getArea( ) {

**return** super.getArea( ) + length;

}

}