Object Oriented terminology

Overloading vs overriding

Widening vs narrowing

# Advantage of using setters (mutator) and getters (accessor) vs public instance variables:

1. getters and setter **can have validation** in them, fields can't
2. using getter you can **get subclass** of wanted class.
3. getters and setters **are polymorphic**, fields aren't
4. **debugging** can be much simpler, because breakpoint can be placed inside one method not near many references of that given field.
5. they can **hide implementation changes**:

# Abstract Classes

A class must be declared ***abstract*** when it has one or more abstract *methods*. A method is declared abstract when it has a method heading, but no body – which means that an abstract method has no implementation code inside curly braces like normal methods do, **but the implementation must be in the children classes.**

NB: When the body does not need a body (eg a method for an abstract class or interface) the body (what is encased in curly brackets{}) is left out and replaced with a semi colon. Eg public abstract double area();

(Access method) (abstract or not) (class or datatype) (method name)();

# Using interface(inherits) from class and inheriting(extends) from a parent:

class Rectangle extends Shape implements Comparables{

In the section on [Interfaces](https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html), it was noted that a class that implements an interface must implement *all* of the interface's methods. It is possible, however, to define a class that does not implement all of the interface's methods, provided that the class is declared to be abstract. For example,

abstract class X implements Y {

// implements all but one method of Y

}

class XX extends X {

// implements the remaining method in Y

}

In this case, class X must be abstract because it does not fully implement Y, but class XX does, in fact, implement Y

