Book 3

Chapter 2

Classes

A summary

Picking class names

a. Begin the class name with a capital letter
b. Use nouns
c. Avoid using names of Java API classes

Knowing what goes in the class body

- Fields: Variable declarations define the public or private fields of a class.
- Methods: Method declarations define the methods of a class.
- Constructors: A constructor is a block of code that's similar to a method but is run to initialize an object when an instance is created. A constructor must have the same name as the class itself, and although it resembles a method, it doesn't have a return type.
- Initializers: These stand-alone blocks of code are run only once, when the class is initialized. There are actually two types, called static initializers and instance initializers.
- Other classes and interfaces: A class can include another class, which is then called an inner class or a nested class. Classes can also contain interfaces.

Order in which to code:

1. Code all the fields together at the start of the class
2. Construc□tors and then
3. Methods.

If the class includes initializers, place them near the fields they initialize.

If the class includes inner classes, place them after the methods that use them.

Fields, methods, classes and interfaces = members;

Understanding methods

- Define methods of a class (review book 2, chapter 7)
- Understand public vs. private

Understanding visibility

- Methods making use of public/private keywords for accessibility refers to visibility
- *Expose* is sometimes used to refer to the creation of public fields and methods.

- Private fields and methods are sometimes called internal members because they're available only from within the class.

Using Getters and Setters

2 accessors – a get accessor (retrieves a field value) and set accessor (sets a field value);

Accessor Pattern

Provides a consistent way to set or retrieve the value of class fields without having to expose the fields;

Overloading Methods

- Methods within a class with the same name but parameters are different.
- NB that signature is unique
- Anticipate different ways one might want to invoke an object's functions and then provide overloaded methods for each alternative
- Methods return type can't be different
- Names of the parameters are not so important…type of parameters and the order in which they appear.

Creating Constructors

Block of code that is called when an instance of an object is created

Constructor has the following properties:

- No return type
- Constructor name is the same as the class name
- Not members of a class
- Called when a new instance of an object is created

Constructor can throw exceptions if it encounters situations

Creating basic constructors

Provide initial class values when creating the object

Constructor for Actor class

```
public Actor(String first, String last)

{

firstName = first;

lastName = last;

}
```

Instance of the Actor class

Actor a = new Actor("Arnold", "Schwarzenegger");

Constructors can also be overloaded – signatures are to be unique

Calling other constructors

Call another constructor of the same class by using ***this*** keyword

```
public class Actor
{
    private String lastName;
    private String firstName;
    private boolean goodActor;

    public Actor(String last, String first)
    {
        lastName = last;
        firstName = first;
    }

    public Actor(String last, String first, boolean good)
    {
        this(last, first);
        goodActor = good;
    }
}
```

Second constructor calls first constructor

- Sets lastName and first then sets goodActor field

Restrictions applied:

Can only call first statement of a constructor