Java Class Design

Oracle® Certified Professional, Java® SE 7 Programmer

Module 1

Module 1 - Objectives

- Use access modifiers: private, protected, and public.
- Override methods.
- Overload constructors and other methods appropriately.
- Use the instanceof operator and casting.
- Use virtual method invocation.
- Override methods from the Object class to improve the functionality of your class.
- Use package and import statements.

Access Modifiers

- public, protected, private, and package (default).
- public members (variables and methods) are visible to any calling class. This is also known as "The Public API".
- protected members (variables and methods) are visible to classes in the same package, or sub-classes of the declared class.
- package (default). Please note that this is convention… there is no access modifier called package. These members are only visible to classes in the same package.
- private members (variables and methods) are only visible within the class.

Access Modifiers (cont.)

- Access modifiers are applicable to static and instance variables and methods. The class will not compile if applied to a local variable.
- It is a best practice to make instance variables private, or protected (if you expect sub-classes) and provide accessors (getters) and mutators (setters).

Overriding Methods

- A method must be implemented (overridden) if one these conditions exist
  - The class is implementing an interface. If the method is not overridden, the class must be declared as abstract.
  - The class is extending an abstract class, and implementing abstract methods. If all of the methods of the abstract class are not implemented, the class must be declared abstract.
- A method of a superclass may be overridden to provide additional specialization in the subclass.
- A class which contains a final method can sub-classed, but its sub-class can not override the final method in super-class.

Overriding Methods (cont.)

- You can not override static, or final methods.
- The access level on the overriding method can not be more restrictive than the overridden method, but it may be less restrictive.
- It must have the same arguments, and return type.
- The return type may be "covariant". This means that the return type may be a subtype of the original return type.
- It may throw any unchecked (runtime) Exception which may not be thrown in the super-class method.
- It may throw more specific exceptions than declared in the superclass, e.g., the super-class throws Exception, and the sub-class throws FileNotFoundException.

Constructor Overloading

- Every class implicitly has a default no-name constructor if no other constructor is provided. it is always called on instantiation.
- A default no argument constructor is provided by the compiler if no other constructor is provided.
- A constructor consists of an access modifier and the class name.
- If a class declares a constructor which takes arguments, the sub-class must provide a constructor that takes the same arguments, or call super() with the same arguments.
- Constructors may take any number of arguments, and may call this() with the same argument list to daisy chain construction.

Constructor Overloading (cont.)

- A call to super(), or this() must be the first call in the constructor.
- super() is called implicitly if not defined.
- Only one call to super(), or this() may be in any constructor.

Method Overloading

- Overloaded methods must change the argument list
- It can have the same, or different return types.
- It can different access modifiers
- It can declare new, or broader exceptions.
- It may be overridden in the same class, or sub-classes.
- The reference type determines which overloaded method is invoked, not the object type.

instanceof Operator

- It will cause a compiler error if the comparison is done with objects which are not in the same class hierarchy.
- Returns true if the type could be cast to the reference type without causing a ClassCastException, otherwise it is false.
- The operand can be a reference type, or null.

Casting

- In order to use an object where it is hidden by a super-class reference, you must cast the object to the sub-class.

```
List list = new ArrayList();
((ArrayList)list).ensureCapacity(10);
```

Virtual Method Invocation

- All method calls are virtual method calls in Java, unless they are static.
- Virtual method invocation is the technical term for determining which method to invoke based on its class hierarchy. For example, If you call eat(); on the Animal interface. The concrete implementation of the method is called. It may be Dog.eat(); or Cat.eat();

Override methods of *Object* Class

- The Object class contains four methods which are often overridden:
  - clone();
  - equals();
  - hashCode();
  - toString();
- The most common method which is overridden is the toString() method. This should be a representation of the class.

Packaging and import Statements

- Packaging is used for grouping of like classes, interfaces, and enums. This may include ancillary classes which are used by the public API.
- import statements are used to differentiate classes which may have the same name, but may have different methods.
- Classes in the same package do not need to be imported.
- Classes in the java.lang.* package do not need to be imported.§7.3

References

- Java™ Language Specification, Java SE 7 Edition, James Gosling, Bill Joy, Guy Steele, Gilad Bracha, and Alex Buckley.
- Preparation for Java Programmer Language Certification
- Sun® Certified Java™ Programmer for Java™ 6 Study Guide, Kathy Sierra, and Bert Bates

creative commons

Attribution-NonCommercial-ShareAlike 3.0 Unported