WSA | Forward looking IT finishing school

Java Programming Language SE – 6

Module 2 : Object-Oriented Programming

ORACLE

Certified Professional

Java SE 6 Programmer

Objectives

- Define object modeling concepts: abstraction,encapsulation and packages
- Discuss why you can reuse Java technology application code
- Define class, member, attribute, method, constructor, and package
- Use the access modifiers private and public as appropriate for the guidelines of encapsulation
- Invoke a method on a particular object
- Use the Java technology application programming interface (API) online documentation

Relevance

- What is your understanding of software analysis and design?
- What is your understanding of design and code reuse?
- What features does the Java programming language possess that make it an object-oriented language?
- Define the term object-oriented.

Software Engineering

The Analysis and Design Phase

- Analysis describes what the system needs to do:

  Modeling the real-world, including actors and activities, objects, and behaviors

- Design describes how the system does it:
  - Modeling the relationships and interactions between objects and actors in the system
  - Finding useful abstractions to help simplify the problem or solution

Abstraction

- Functions – Write an algorithm once to be used in many situations
- Objects – Group a related set of attributes and behaviors into a class
- Frameworks and APIs – Large groups of objects that support a complex activity; Frameworks can be used as is or be modified to extend the basic behavior

Classes as Blueprints for Objects

- In manufacturing, a blueprint describes a device from which many physical devices are constructed.
- In software, a class is a description of an object:
  - A class describes the data that each object includes.
  - A class describes the behaviors that each object exhibits.

Classes as Blueprints for Objects

- In Java technology, classes support three key features of object-oriented programming (OOP):
  - Encapsulation
  - Inheritance
  - Polymorphism

Declaring Java Technology Classes

- Basic syntax of a Java class:

```
<modifier>* class <class_name> {
<attribute_declaration>*
<constructor_declaration>*
<method_declaration>*
}
```

Declaring Java Technology Classes

```
public class Vehicle {
private double maxLoad;
public void setMaxLoad(double value) {
maxLoad = value;
}
}
```

Declaring Attributes

- Basic syntax of an attribute:

```
<modifier>* <type> <name> [ = <initial_value>];
```

- Examples:

```
public class Foo {
private int x;
private float y = 10000.0F;
private String name = "Bates Motel";
}
```

Declaring Methods

Basic syntax of a method:

```
<modifier>* <return_type> <name> ( <argument>* ) {
<statement>*
}
```

Declaring Methods

```
public class car{
 private int modelno;
private String colour;

public void dispInfo(int mno,String col) {
modelno=mno;
colour=col;


}
}
```

Declaring Methods
Example-2

```
public class Dog {
private int weight;
public int getWeight() {
return weight;
}
public void setWeight(int newWeight) {
if ( newWeight > 0 ) {
weight = newWeight;
}
}
```

Accessing Object Members

- The dot notation is: <object>.<member>
- This is used to access object members, including attributes and methods.
- Examples of dot notation are:
- a.dispInfo(101,”green”);
- a.modelno=”101”;

Example

// create a class car that will display the model number and colour of car.

```
class car
{
   int modelno;
   String colour;

    void dispInfo(int mno,String col)
    {
         modelno =    mno;
         colour     =     col;
     }
}
```

Example -2

```
class  carTest

{
        car c1=new car();
       c1.dispInfo(101,”green”);
}
}
```

Encapsulation

- Hides the implementation details of a class
- Forces the user to use an interface to access data
- Makes the code more maintainable

Car	
model_no	int
colour	String
disp_info()	void
Move()	void

Declaring Constructors

- Basic syntax of a constructor:

```
[<modifier>] <class_name> ( <argument>* ) {
<statement>*
}
```

- Example:

```
public class Car {
private int speed;
public  Car() {
speed = 50;
}
}
```

The Default Constructor

- There is always at least one constructor in every class.
- If the writer does not supply any constructors, the default constructor is present automatically:
  - The default constructor takes no arguments
  - The default constructor body is empty
- The default enables you to create object instances with new Xxx()without having to write a constructor.

Source File Layout

- Basic syntax of a Java source file is:

```
[<package_declaration>]
<import_declaration>*
<class_declaration>+
```

Source File Layout

```
package shipping.reports;

import shipping.domain.*;
import java.util.List;
import java.io.*;

public class VehicleCapacityReport {
private List vehicles;
public void generateReport(Writer output) {...}
}
```

Software Packages

- Packages help manage large software systems.
- Packages can contain classes and sub-packages.

The package Statement

- Basic syntax of the package statement is: package
  - <top_pkg_name>[.<sub_pkg_name>]*;
- Examples of the statement are:
  - package shipping.gui.reportscreens;
- Specify the package declaration at the beginning of the source file.
- Only one package declaration per source file.
- If no package is declared, then the class is placed into the default package.
- Package names must be hierarchical and separated by dots.

The import Statement

- Basic syntax of the import statement is:

  Import<pkg_name>[.<sub_pkg_name>]*.<class_name>;

  OR

  import<pkg_name>[.<sub_pkg_name>]*.*;

- Examples of the statement are:

  import java.util.List;

  import java.io.*;

  import shipping.gui.reportscreens.*;

The import Statement

The import statement does the following:

- Precedes all class declarations
- Tells the compiler where to find classes

Compiling Using the -d Option

```
cd JavaProjects/ShippingPrj/src
javac -d ../classes shipping/domain/*.java
```

Recap

- Class – The source-code blueprint for a run-time object
- Object – An instance of a class;

  also known as instance
- Attribute – A data element of an object;

  also known as data member, instance variable, and data field
- Method – A behavioral element of an object;

  also known as algorithm, function, and procedure

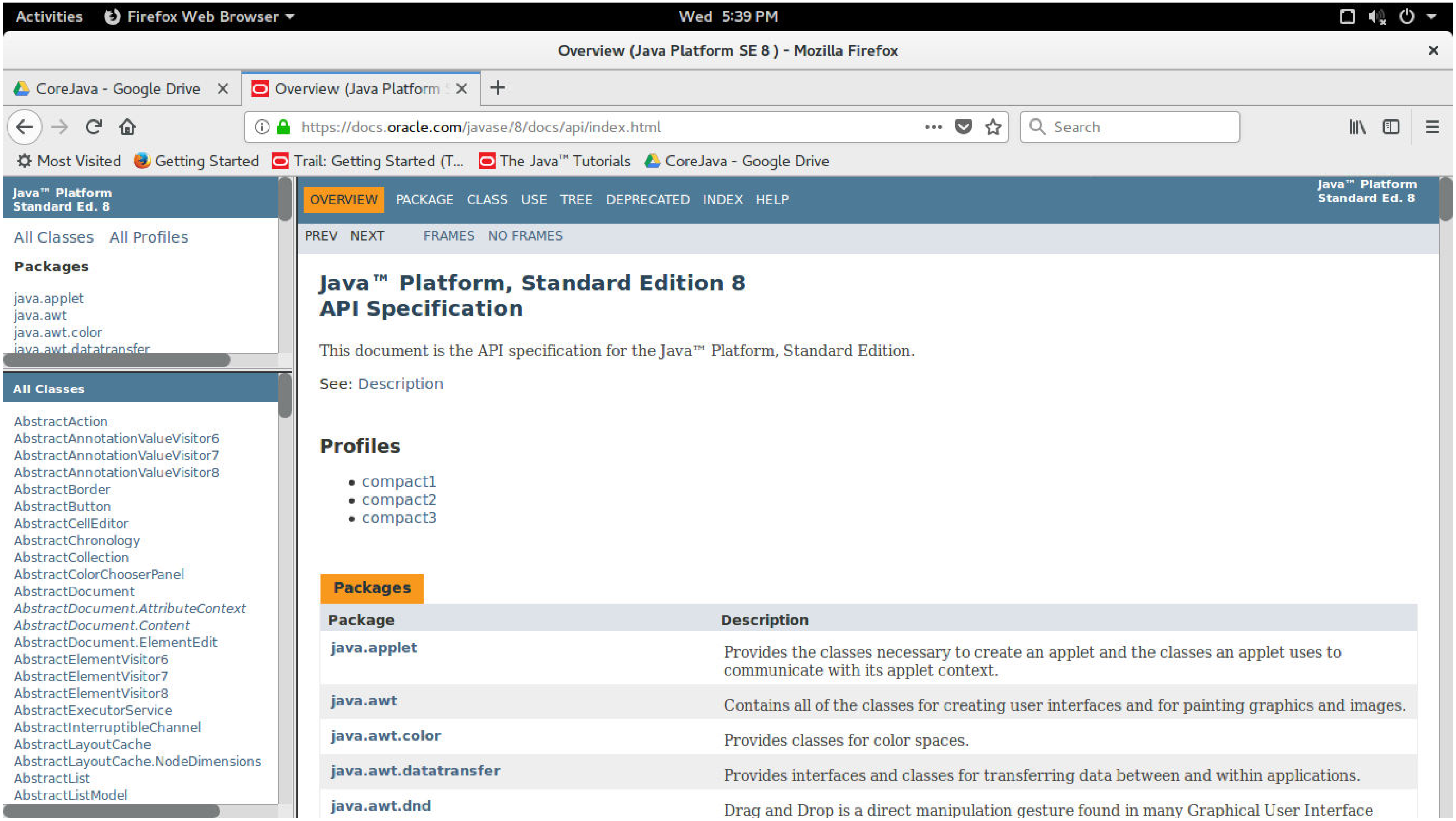
Recap

- Constructor – A method-like construct used to initialize a new object
- Package – A grouping of classes and sub-packages

Java Technology API Documentation

- A set of Hypertext Markup Language (HTML) files provides information about the API.
- A frame describes a package and contains hyperlinks to information describing each class in that package.
- A class document includes the class hierarchy, a description of the class, a list of member variables, a list of constructors, and so on.

Java Technology API Documentation



Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer

All Classes

AbstractAction
AbstractAnnotationValueVisitor6
AbstractAnnotationValueVisitor7
AbstractAnnotationValueVisitor8
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractChronology
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeContext
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractElementVisitor6
AbstractElementVisitor7
AbstractElementVisitor8
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDimensions
AbstractList
AbstractListModel

Java™ Platform, Standard Edition 8 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Profiles

- compact1
- compact2
- compact3

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface

Thank You