This page has examples of chapter opener bullets. We prefer gerunds (-ing words). The next pages give examples of chapter summary bullets.

Foundation: XML and JSP

This chapter covers

- The basics of XML syntax
- An introduction to JSP
- JSTL's tag libraries in context
- JSP scoped variables

Working with XML fragments

This chapter covers

- Referring to XML documents with XPath variables
- Printing parts of XML documents
- Loops and conditions with XML documents
- Invoking XSLT transformations from your pages

Case study in building a website

This chapter covers

- Designing a reusable layout
- Plugging modular channels into a web site
- Registering and authenticating users
- Personalizing a web site

This page and the following pages give examples of good chapter summary bullets.

1.4.3 Newsgroups

The principal newsgroups for discussions of object-oriented programming are comp.object and comp.object.moderated. As usual, the unmoderated group has much greater level of traffic and more noise, while the moderated group drifts into deep and theoretical discussions at times. However, both groups have high signal-to-noise ratios and are relatively novice-friendly.

1.5 SUMMARY

- Objects are mechanisms that provide controlled access to collections of data (attributes) and allow them to be manipulated in predefined ways.
- This access and manipulation is provided by methods, which are subroutines specifically associated with a particular class of objects.
- A class defines the attributes and methods that a certain type of object provides.
- Inheritance allows new classes of objects to be created by extending or altering the behavior of an existing class. A derived class has all the attributes and methods of the classes it inherits and typically adds new ones as well.
- The way an object responds to a standard method call often depends on the kind of object it is. This is known as polymorphism.
- The main advantage of object orientation is that it separates the interface to data from the implementation of that data and of the operations defined on it.
- Object attributes may also be objects of simpler classes. Building larger objects by joining together smaller ones is known as aggregation.
- Sometimes it's useful to generalize the syntactic structure of a set of similar classes, rather than their semantics. Such structural generalizations produce generic classes or methods.

For more information on various one-way encryption schemes such as `crypt`, MD5, PGP and SHA (also known in cryptography circles as message digests or hashes) see http://dir.yahoo.com/Computers_And_Internet/Security_And_Encryption/Cryptography/.

4.5 SUMMARY

- Hashes are the usual basis of objects. When in doubt, use a hash.
- Arrays generally provide better access speed. Use constants to give attributes logical names.
- Arrays may also be a better choice if the objects being implemented have a fundamentally listlike structure or function.
- Pseudo-hashes provide the convenience of hash-like access with the performance of array-like access, but only if we use typed lexicals to access them.
- Occasionally, an object may be most efficiently implemented as a blessed scalar.

Chapter summary bullet examples from *Unity in Action*

You can use the examples on this page and the next as follows:

1. If you already have chapter summary bullets, but your DE has asked you to revise them, use these examples for ideas.
2. The “After” examples are more good examples (in addition to the previous pages) of Manning’s bulleted summaries.

Chapter 1

Before

- Unity is a multiplatform development tool.
- Unity’s visual editor has several sections that work in concert.
- Scripts are attached to objects as components.
- Code is written inside scripts using MonoDevelop.

After

- Unity is a development environment optimised for developing games
- The interface in Unity is split into six different sections. The Scene View shows the game world and allows you to move objects around, the Game View shows the game when running, the Toolbar has controls for working with the scene, the Hierarchy tab allows you to drag and drop object relationships, the Inspector display information about the selected object(s), and the Console tab displays any messages from your code - debugging, status, etc - while the game is running.
- All code execution in Unity starts from code files linked to an object in the scene.
- "Doing" Scripts inherit from MonoBehaviour which does all the linking and provides common methods like Start() and Update()
- Script files can be written in either Unity or MonoDevelop.

Chapter 2

Before

- 3D coordinate space is defined by X-, Y-, and Z-axes.
- Objects and lights in a room set the scene.
- The player in a first-person scene is essentially a camera.
- Movement code applies small transforms repeatedly in every frame.
- FPS controls consist of mouse rotation and keyboard movement.

After

- Building a basic game scene requires you to define and place floors, walls, lights and a camera.
- To view a scene from a first person perspective, you create a player object and attach a camera to it. Attaching movement scripts to the player object allows you to rotate the camera and move it around with mouse and keyboard.

- Objects can be positioned in the scene by typing 3D coordinates (x,y,z) into the Inspector window. The size of an object is defined by its scale.
- Every scene object inherit from GameObject which knows how to attach scripts that control it (the object)
- There are three types of light : rays from point lights originate from a single location and radiate in all directions, rays from spot lights do as point lights but radiate in a defined cone and rays from directional lights are parallel and project evenly across a scene (like the sun).
- A minimal movement script means calling Translate, Rotate or Scale at least once during the object's Update method. This call is then performed every time the game scene is updated (every 'frame').
- Use Input.GetAxis to identify and respond to mouse movements and key presses.
- The Rotate method spins an object. It can use either local coordinates (the default) or global coordinates. Local coordinates are defined relative to the object being moved while global coordinates are defined relative to the entire game scene.
- The Translate method changes the coordinates of an object within a game scene. Use it to define how a player object moves around and the CharacterController object's Move method to enable collision detection and make sure the player isn't walking through any other object.