

1. Chapters 1 & 2

a) (2 pts.) Answer by placing a Y (yes) or N (no) immediately after the following questions:

- The UML is a process to guide the development of object-oriented software
- An object-oriented system is characterized as a set of communicating classes
- A sequence diagram defines the objects that we are interested in when describing the use-case
- An object of a class can send a message to an other object of the same class
- Sequence diagrams and collaboration diagrams convey essentially the same information
- An actor provides an internal stimuli for a use-case
- Lifelines in a sequence diagram are used to show the when objects are active or idle
- An activity diagram is often used when modeling various static aspects of a system

b) (1 pts.) Explain why a class diagram is considered an abstract representation, while an object diagram is considered a concrete representation. Illustrate the difference with an example.

c) (1 pt.) Discuss four properties of composite aggregation of objects

2. Chapter 3

a) (4 pts.) Answer by placing a Y (yes) or N (no) immediately after the following questions:

- UML mandates a particular software development lifecycle
- In a lightweight process there is a single iteration
- In a Java class declaration, we use **void** to indicate that return value is zero
- Java prescribes a particular ordering of the features of a class
- An object diagram is the same as a collaboration diagram without any messages
- A collection object is a container for messages between objects
- The method **main** in Java is a **static** method
- The Application class is not associated with any other class

b) (1 pt.) What is meant by information hiding when referring to a class declaration? Name two visibility levels provided by Java for implementing information hiding.

c) (1 pt.) What does the mandatory profile specify? Give the mandatory profile for the bank application presented in Chapter 3 of your textbook

d) (1 pt.) Give a Java code, with two sample methods, for the class Point, representing a point in the Cartesian co-ordinate system.

3. Chapter 4

a) (2 pts.) Answer by placing a Y (yes) or N (no) immediately after the following questions:

- In the Library application, the **private** method **testUseCase** in the **Application** class executes “hard-wired” versions of a pre-defined set of test-cases
- An attribute in a UML class maps to a public field declaration in a Java class
- The use of a collection in UML maps to an **implements** statement naming the class of the collection in Java
- The % symbol associated with classes in a package means that they are not available to clients but have visibility of each other
- An association or aggregation in UML maps to a public field declaration in Java
- The operation **toString** is defined in the class **Object**
- Each use-case should be accompanied by at least one test-case
- The domain model should have responsibility for any input and output

b) (1 pt.) Give the Java code (6-8 lines) for a class Application that includes testing for 2 possible Error Conditions in “registering a Borrower” in the Library Application

c) (1 pt.) Explain the role of the Action class in general, and discuss its relationship with the Application class and the Domain model.

4. Chapter 5

a) (2 pts.) Answer by placing a Y (yes) or N (no) immediately after the following questions:

- If descendant class substitution is combined with late binding then the polymorphic effect is the result
- An abstract class must have at least one subclass
- A subclass does not have to respond to the same messages as the parent
- The concept of polymorphism does not apply to constructors
- A class that implements an interface must define its inherited methods or it is itself abstract.

- A class is abstract when one or more of its methods is qualified as abstract
- It is an error to define a protected or private method in an interface
- An interface cannot define attributes

b) (1 pt.) Can a class implement more than one interface? Explain and illustrate it with an example

c) A class diagram consists of several symbols. They may be either a class symbol or a relation symbol. With the former the symbol may describe a concrete, abstract, or interface class. With the latter the symbol may describe a shared aggregation, composite aggregation, or specialization relation. A relation symbol always connects two class symbols, and a class symbol may have zero or more relation symbols connected to it.

(2.5 pts.) Construct a class diagram for the ABOVE DEFINITION of the class diagram object. I'm asking you to model, using a class diagram, the definition of a class diagram as given above; I am NOT asking you to give a class diagram for a sample application!

d) (1 pt.) Abstract classes and interface classes have similar features, and it is not always clear when to use one over the other. Explain under what circumstances it makes sense to use both.

5. Chapter 6

a) (1 pt.) Answer by placing a Y (yes) or N (no) immediately after the following questions:

- Information hiding gives rise to the notion of polymorphism
- Usually specialization and aggregation relationships are combined in the same diagram
- In the Library Application, the Publication class has the common features of Books and Journals
- **this.getPublication ()** means obtain a reference to a new Publication object

b) (0.5 pt.) Explain in words what the following line of Java code is doing:
“**public abstract class** Publication **implements** Comparable”

c) (0.5 pt.) Is it true or false that when a reference to an interface is declared, only messages corresponding to operations advertised by that interface can be sent through that interface? Explain.

d) 0.5 pt.) Explain in words what the following line of Java code is doing:
“**public final class** Book **extends** Publication”

e) (1 pt.) Explain persistence of an object in Java. Name the classes used, and give 2-3 lines of code illustrating the persistence of the object Book

6. Chapter 7

a) (2 pts.) Answer by placing a Y (yes) or N (no) immediately after the following questions:

- The abstract class JComponent is the root class for many of the graphic components
- Components can not include other sub-components in a parent/child arrangement.
- JFrame is a general-purpose container frequently used to group other graphical components
- The AWT library preceded the development of the Swing library
- The Java event model is based on the notion of event listeners.
- When you want the look and feel of your graphics application to compatible with older Web browsers, you should choose AWT over the Swing package in building your GUI
- Inner classes are rarely used to realize event listeners
- The class Container is a specialization of the Swing class JComponent

b) (1 pt.) What is the rationale for adding ActionListener objects to both JMenu objects and to JToolBar objects?

c) (1 pt.) What design pattern(s) were used in defining the Java event model. Describe what happens when an event occurs.

d) (1 pt.) Explain in words what the following line of Java code is doing:
`LibraryFrame.this.theFileExitAction.actionPerformed(null);`

Chapter 8

- a) (0.5 pt.) Draw the Model-View-Controller (MVC) inspired architecture for the Library Application, clearly identifying what classes belong to model, view, and controller
- b) (0.5 pt.) Define the decorator design pattern. Does it apply to the textbook's Library Application. Explain
- c) (0.5 pt.) Define the visitor pattern. Does it apply to the textbook's Library Application. Explain
- d) 0.5 pt.) Define the observer pattern. Does it apply to the textbook's Library Application. Explain
- e) (0.5 pt.) Define the template method pattern, and give an example of its use in the Library Application

8. What's new in JavaLand

- a) (2 pts.) Name, discuss, and give a piece of code illustrating each of 4 new language features in Java SE 5.0
- b) (0.5 pt.) Name 2 new features in Java SE 6.0