

11

*The Programming
Process*

Self-review Questions

11.1 *What is a problem domain? Why are problem domains relatively stable?*

When identifying the requirements for a software application a description of the real-world environment in which the software will operate is needed. This is the problem domain. It will describe the concepts, entities, relationships and tasks that take place. The application being developed will either interact or replace some of these elements.

A description of a problem domain typically includes a vocabulary, a list of entities and a set of task descriptions.

11.2 *Identify the problem domain entities for payroll systems.*

A very basic approach is to write a description of how the payroll process is currently managed and run within a specific organisation and then look for the things named. Input to and output from the payroll process should be carefully identified. Note, however, doing this for real would be a much more rigorous and detailed process, making use of Software Engineering techniques like requirements analysis and use cases.

Example entities might include: Employee, payslip, money, tax, deduction, salary, date, hours worked, pay rate, pay clerk, payroll run, bank, bank account and so on. Not all of these would be part of a software payroll system but are relevant as they need to interact with such a system.

11.3 *What are the strengths and weaknesses of prototyping?*

A prototype is often considered as a throw-away piece of code that has served its useful purpose as identifying the correct design for an application but is not of good enough quality to actually be the production version of the application. A prototype should be developed only as far as necessary to identify a good solution.

Strengths:

- Demonstrates the feasibility (or not) of building a working application based on the current specification.
- Allows the design and implementation to be explored.
- Can make rapid progress in finding out how to solve a problem without the overhead of needing to create production quality code.

Weaknesses:

- What is meant to be experimental code becomes production code due to time and resource constraints, resulting in poor quality code and design.
- The prototype becomes too big and complex, using up too much time and resources to develop.

11.4 *Write a collection of scenarios for an application to manage the renting of videos from a video rental store.*

11.5 *What is the role of analysis?*

Analysis is about identifying the required behaviour of an application that is to be developed. This typically involves specifying the requirements in detail, identifying the data the application will work on and specifying the tasks that need to be performed showing how data is input, transformed and output.

With Object-Oriented Analysis (OOA), the behaviour is modelled in terms of objects and their collaborations (method calls on each other).

11.6 *How can classes be identified?*

Identifying a good set of classes is a key part of object-oriented design. One way to start is to identify entities, strategies and concepts by systematically reviewing the requirements specification. This activity can be initialised by looking at the nouns in the requirements and treating the named things as potential classes.

Once a small set of potential classes have been identified, the Class, Responsibility, Collaboration method (CRC — see Appendix B in *Developing Java Software*) can be used to identify attributes and methods for the proposed classes, along with the relationships between classes. In addition, new classes are likely to be identified and some of the potential classes dropped as they turn out not to be relevant.

With a working set of classes, attributes and methods, a process like Test-driven Development (TDD) can be started to turn the proposed classes into code. This is highly likely to lead to the further addition and removal of classes as the design falls into place. It is also possible to use (TDD) right from the beginning as a way of identifying classes

11.7 *What is the role of design?*

Design is about determining how a software application will be realised. It involves identifying and specifying packages, detailed classes, data structures, methods, algorithms and all the details needed to actually go ahead and implement an application.

Good design is typically an exploratory process with coding and testing forming an important part of determining whether a design will work and if it is a good solution for the current context. Attempting to completely design an application before implementation and testing invariably leads to serious difficulties as it is very hard to validate a design on its own.

Design can be viewed as a step within the development process but the label 'design' is best applied to the entire activity of specifying and creating a working implementation.

11.8 *List the properties of a well designed class.*

A class should:

- Be cohesive, meaning that it should focus on representing one entity, concept or strategy. A class that is a collection of several distinct parts should be split up.
- Not be too long (too many lines of code) or have too many methods. A class that gets too long is probably not very cohesive and should be split up.
- Make proper use of encapsulation, so that the internal implementation and data structures are not exposed to users of the class. This reduces the coupling or dependency of client code on a specific class implementation. Instance variables should always be private or protected.
- Have a thin interface, meaning that the minimal set of public methods should be available.

11.9 *What is the aim of testing and what are the limitations?*

The aim of testing is to find errors, so tests should be focussed on those most likely to locate an error. Exhaustive testing, that is testing every possible state, is not usually possible and, in addition, testing is subject to time and budget constraints. Hence, tests should be selected to maximise the chance of errors being found. Although developers often talk of 'all tests passing' meaning that no errors were found, a successful test is better characterised as one which finds an error, as that error can then be removed from the software.

Tests that do not find any errors give confidence that the code works according to the specification set by the tests but cannot demonstrate that no errors exist. Testing can not prove the absence of errors in the formal sense, and the major limitation of testing is that undetected errors can still exist as no test has been written to trigger them.

11.10 *Make suggestions as to how a code review might be conducted.*

First and foremost, a code review should not be a confrontation. The aim is to locate errors and make suggestions for improvements to the code quality.

11.11 *What is an appropriate amount of documentation for a program? What should it include?*

11.12 *What is the key information that you require in order to effectively maintain code written by someone else?*

11.13 *Summarize the strengths and weaknesses of object-oriented development.*

11.14 *What is the point of reuse? What features should a reusable class have?*

Programming Exercises

11.1 *Write Java classes corresponding to the classes shown in this UML diagram.*

11.2 *The UML class diagram below represents a simple order system.*

Write a Java class for each class identified in the diagram.

Challenges

11.1 *Consider the following specification:*

"A program is required to run the controller of a burglar alarm system. A typical system consists of a number of sensors connected by individual circuits to a central control box containing the controller. The control box has a simple keypad and display. Sensors include

switches, heat detectors and motion detectors. Each sensor has an identification code which can be read by the controller to identify the sensor.

The controller allows an operator to select which sensors are active and turn on or off the system. If a sensor is triggered when the system is active, the controller must activate the alarms (a siren and a bell) and display a message on the display panel indicating which sensor is involved. The operator must enter a security code before the system is turned on or off."

1. *Identify a set of classes which might be used to model the system from an object-oriented point of view.*
2. *Use the classes to construct a class diagram to show the structure of the system.*
3. *Write a program to simulate the system, using the classes.*

11.2 *Create a UML class diagram showing the classes and their relationships in the JDK package `java.util`.*