Chapter 12

Programming and Testing

Topics

- Quick tutorial in Java from software engineering viewpoint
  - Class
  - Class associations and collections
  - Database access in Java
- Test-driven development
- Acceptance and regression testing
Java class

- its name and visibility
  - class visibility is normally declared as public
- member variables (data members)
  - instance variables and class variables (static)
- member functions (methods)
  - instance methods and class methods (static)
- constructors
- visibility of members
  - can be public, protected, private, or package
  - class interface (protocol) - set of all public methods of a class
- its superclass (if any) and interfaces (if any)

Java class in UML

```java
Movie

movieTitle : String
movieCode : double
director : String

Movie(movieCode : double, title : String, director : String)
addListedAs(l : ListedAs) : void
removeListedAs(l : ListedAs) : void
getMovieTitle() : String
setMovieTitle(title : String) : void
getMovieCode() : double
setMovieCode(code : double) : void
getDirector() : String
setDirector(dir : String) : void
equals(o : Object) : boolean
```
Definition (implementation) of Java class

Movie.java

```java
public class Movie {
    private String movieTitle;
    private double movieCode;
    private String director;
    private Collection listedAs;

    public Movie(double movieCode, String title, String director) {
        this.movieCode = movieCode;
        this.director = director;
        this.movieTitle = title;
        listedAs = new ArrayList();
    }
    ...
}
```

Java collections

- `Collection` interface represents a container for objects (called elements).
- Its methods allow adding new elements, removing existing elements, finding if it contains a specific element, checking if the collection is not empty, checking its size (the number of elements), placing the elements in an array, and iterating over its elements using an `Iterator` object.
- `java.util.Collection` library delivers three categories of containers:
  - List
  - Set
  - Map
From conceptual...

```
actor
  actor_code : double
  actor_name : String

movie
  movie_code : double
  movie_title : String
  director : String

listed_as
  position : double
```

...to design class model

```
ListedAs
  position : double

ListedAs()
  getPosition()
  setPosition()
  getMovie()
  setMovie()
  getActor()
  setActor()

Actor
  actorCode : double
  actorName : String

Actor()
  addListedAs()
  removeListedAs()
  getActorCode()
  setActorCode()
  getActorName()
  setActorName()
  equals()

Collection (from util)

Movie
  movieTitle : String
  movieCode : double
  director : String

Movie()
  addListedAs()
  removeListedAs()
  getMovieTitle()
  setMovieTitle()
  getMovieCode()
  setMovieCode()
  getDirector()
  setDirector()
  equals()```

Actor and Movie contain:

```
listedAs: Collection
```
**JDBC**

- **JDBC** (Java Database Connectivity) is an API (Applications Programming Interface) which allows Java to send SQL statements to a relational database
  - it uses a JDBC driver to connect to a database
- Use JDBC to do dynamic SQL
  - Dynamic SQL means that the embedded SQL statement to be executed is not known before the application is run, and requires input to build the statement
- Once a Java program loads a database driver, a database connection (Connection object) can be established
- JDBC statements and calls to stored procedures can then be created (Statement objects) and executed on that connection

```java
// Query the employee names
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT ENAME FROM EMP");
```

**SQLJ**

- A language specification for embedding static SQL statements in Java source code
- SQLJ runs on top of JDBC and requires a translator that replaces the embedded SQL statements with calls to SQLJ runtime environment
  - The translator is a preprocessor that converts a SQLJ source program (with a .sqlj extension) into a Java source (.java extension), before compiling the source to produce a class file (.class extension)
- JDBC and SQLJ code can be mixed in a program sharing the same connection and result sets

```java
String name;
@sql1 { SELECT ename INTO :name FROM emp WHERE empno=67890 };
System.out.println("Name is "+name+", employee number = "+empno);
```
### SQLJ design goals

- The primary goal is to provide simple extensions to Java to allow rapid development and easy maintenance of Java applications that use embedded SQL to interact with databases.
- Provide a concise mechanism for database access via static SQL
  - Most SQL in applications is static
  - Check static SQL at translate time
- Provide flexible deployment configurations
  - This makes it possible to implement SQLJ on the client or database side or in the middle tier
- Support a software standard
  - SQLJ is an effort of a group of vendors and will be supported by all of them
  - Applications can access multiple database vendors
- Provide source code portability
  - Executables can be used with all of the vendors' DBMSs presuming the code does not rely on any vendor-specific features

### Comparison of SQLJ with JDBC

- JDBC code and SQLJ code interoperates, allowing dynamic SQL statements in JDBC to be used with static SQL statements in SQL
  - A SQLJ iterator class corresponds to the JDBC result set.
- SQLJ source code is more concise than equivalent JDBC.
- SQLJ uses database connections to type-check static SQL code. JDBC, being a completely dynamic API, does not.
- SQLJ provides simplified rules for calling SQL stored procedures and functions.
- SQLJ programs allow direct embedding of Java bind expressions within SQL statements. JDBC requires a separate get and/or set call statement for each bind variable and specifies the binding by position number.

```java
while (rset.next())
    System.out.println(rset.getString(1));
```
**ODBC**

- Microsoft’s ODBC (Open DataBase Connectivity)
  - ability to connect to almost all databases on almost all platforms

- ODBC uses a C interface
  - Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness, and automatic portability of applications

- ODBC is harder to learn (in particular for a Java programmer who does not need to worry about pointers (address variables), memory management, data byte alignment, etc.)

- A Java API like JDBC is needed in order to enable a “pure Java” solution so that the code is automatically installable, portable, and secure on all Java platforms from network computers to mainframes

- JDBC-ODBC bridge

---

**Testing types and development phases**

- Programming
  - Design specifications
  - Analysis specifications
  - System constraints
  - Customer requirements
  - User environment

- Unit test

- Reusable components
  - Integration test
  - Functional test
  - Constraint test
  - Acceptance test
  - Installation test

- System deployed

- Performance, Security, Scalability, etc.
The essence of test-driven

- Invites developers to write test specifications and programs before deciding on the final design and before starting “cutting the application code”.
  - The application code is written as a response to a test code, not vice versa.
  - If a test code is written before the application code, then clearly the test program will fail when run.
    - That is precisely the point. A test code is written to fail the application code.
  - The application code should be implemented so that the test will succeed next time it is run.
    - This will assert the existence of the functionality demanded by the test.
  - The essence of test-driven development is to drive the software development, not the software verification.

Test units and suites

- Test-driven development starts as a unit-based testing.
- Test unit has two interpretations:
  - a target of the test
    - a single class in the application code, but it can also be just selected methods of a class or a few cooperating classes
  - a resource of the test
    - a piece of code – a class or a set of classes that perform the testing
    - methods within a test unit class are called test cases
- Test cases can be combined in test suites to run a collection of test cases that target many classes or even the whole system.
Test code vs application code

- Where to place the test code with regard to the application code to be tested?
  - within a main() method of each class to be tested (i.e. within each test unit)
  - in a static inner class within a class to be tested
    - since in Java, an inner object can freely access all elements of an enclosing outer object, it can be advantageously used to contain a test case
  - using a generic testing framework
    - libraries of classes and interfaces aimed at facilitating test implementations
    - JUnit is a Java framework recommended by the agile software development and XP in particular

Test-driven example

Method testLogin() in CActionerTest

```java
void testLogin()
    CActioner actioner = new CActioner();
    try{
        IAEmployee emp = actioner.login("user","passwd");
        assertEquals(emp.getLoginName(), "user");
    }catch(Exception exc){
        fail("Exception occurs during login");
    }
```

Method login() in CActioner

```java
private IAEmployee emp;
...
public Object login( String username, String passwd ) {
    emp = broker.login( username, passwd );
} 
```
Acceptance testing

- **Black-box tests**, which verify if the use case requirements are met, with no consideration given to the internal workings of the software
- Written as **test scripts**
  - an expectation is that most of these test scripts will be automated by using capture/playback tools
  - test scripts that cannot be automated will be used for manual testing by a human tester
- As opposed to test-driven
  - a unit of an acceptance test is a piece of functionality that normally spans multiple classes
  - acceptance tests contain **verification points**, which check if the expected functionality is met by the implementation

Summary

- The definition of Java class consists of the class name and visibility, data members and methods, constructors, members’ visibility, and any class’ superclasses and interfaces
- Java collection represents a container for objects – can store association links with the many multiplicity
- JDBC and SQLJ are two standard ways to embed SQL statements in Java code in order to communicate with a database
- Test-driven development requires writing test programs prior to writing the application code to pass the test program
- Acceptance testing is used to test application code after it has been written