Multiple-Choice Quiz

This section helps you review what you have learned.

1. If a class contains a constructor, that constructor will be invoked
   (a) once the first time an object of that class is instantiated
   (b) each time an object of that class goes out of scope
   (c) each time an object of that class is instantiated
   (d) once at the beginning of any program that uses that class
2. What is used to indicate that a method does not return a value?
   (a) the omission of the return type
   (b) the keyword static
   (c) the keyword void
   (d) the name of the class to which it belongs
3. Which is the Java keyword used to denote a class method?
   (a) private
   (b) static
   (c) class
   (d) final
4. Which of the following categorizations can be applied to both the data fields and the methods in a Java class?
   (a) default and non-default
   (b) native and non-native
   (c) static and non-static
   (d) abstract and non-abstract
5. Consider the following Java program segment.

```
import java.io.*;
public class Test {
    public Test( ) {
        System.out.println("default");
    }
    public Test( int i ) {
        System.out.println("non-default");
    }
    public static void main(String[] args) {
        Test t = new Test(2);
    }
}
```

   Which of the following will be output during execution of the program segment?
   (a) The line of text "default"
   (b) The line of text "default" followed by the line of text "non-default"
   (c) The line of text "non-default" followed by the line of text "default"
   (d) The line of text "non-default"
6. If the method int sum(int a, int b) is defined in a Java class C, which of the following methods cannot coexist as a different method in class C?
   (a) int sum(int x, int y)
   (b) int sum(int x, float y)
   (c) int sum(float a, int b)

(d) float sum(int x, float y)

7. From within a child class, its parent class is referred to via the keyword
   (a) this
   (b) base
   (c) parent
   (d) super
8. When a subclass defines an instance method with the same return type and signature as a method in its parent, the parent's method is said to be
   (a) hidden
   (b) private
   (c) overloaded
   (d) overridden
9. Consider the following Java class definitions.

```
public class Object1 {
    protected String d(){
        return "Hi";
    }
public class Object2 extends Object1 {
    protected String d(){
        return super.d();
    }
```

   Which of the following statements is (are) true regarding the definitions?
   Class Object2 inherits from class Object1.
   Class Object2 overrides method d.
   Method d returns equivalent results when executed from either class.
   (a) I, II, and III
   (b) III only
   (c) I and II only
   (d) I and III only
10. Which of the following statements is (are) true about all data fields in an interface in Java?
    They are implicitly public.
    They are implicitly final.
    They are implicitly static.
    (a) I and II only
    (b) II only
    (c) I, II, and III
    (d) II and III only

Polymorphism

Recall the second problem in lab4. In this question, you will be able to use method overriding and understand how this gives rise to the feature of dynamic method binding, also known as polymorphism, and understand why it is better to use polymorphism than run-time type enquiry.

Questions:

1. Finish the second problem in lab4 if you haven't already and then load the file

ShapeTest.java into your text editor.

2. The questions of this exercise take you through the steps of reworking the classes of lab4 to use polymorphism rather than run-time type enquiry. You will then see the advantage of polymorphism over run-time type enquiry.

Video Home System and DVD Movies

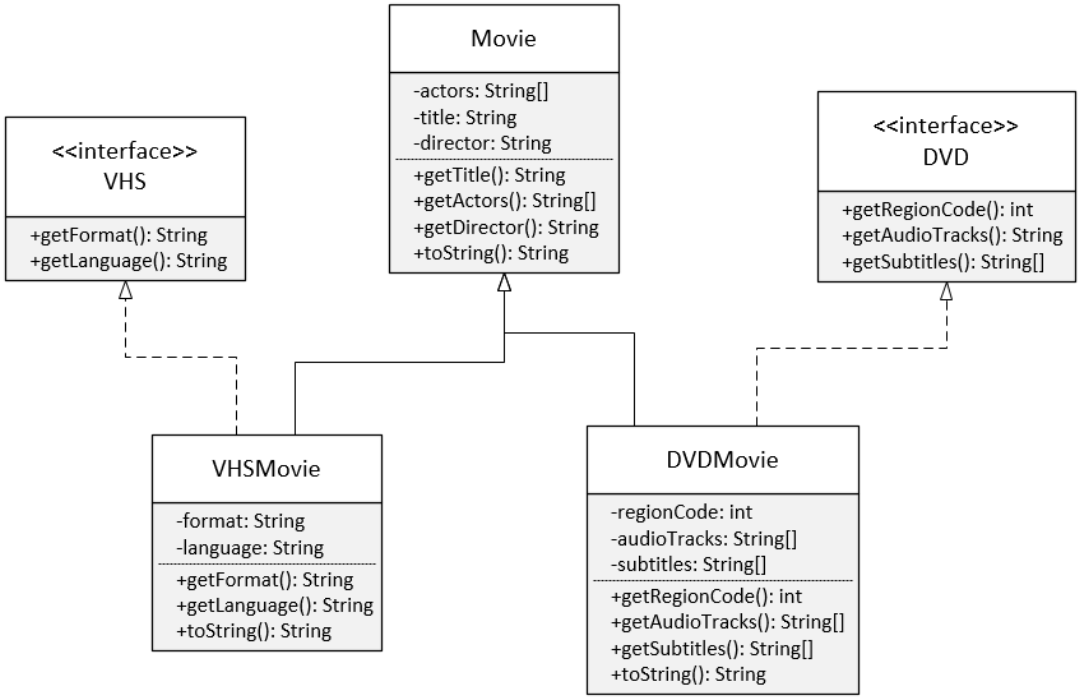
Background

In this assignment, you will create the following classes and interfaces:

- Abstract class (Movie)
- Interfaces (VHS, DVD)
- Classes (VHSMovie, DVDMovie)

Description

The following class diagram illustrates the relationships between the interfaces and classes:



The specification of the interfaces and classes are as follows:

Abstract class Movie

The abstract class Movie stores the information of a movie.

Instance variables:

- String title. The title of the movie
- String[] actors. The names of the actors in the movie
- String director. The director of the movie

Constructor and methods:

- Movie(String initialTitle, String[] initialActors, String initialDirector)

Creates a Movie object and initializes the instance variables.

- String getTitle(). Returns the value of the variable title.
- String[] getActors(). Returns a reference to the array actors.
- String getDirector(). Returns the value of the variable director.

- String toString(). Returns the value of the variable title.

Interface VHS

The interface VHS declares the methods for obtaining VHS tape information.

Methods:

- String getFormat(). Returns the format of the VHS tape.
- String getLanguage(). Returns the language of the VHS tape.

Interface DVD

The interface DVD declares the methods for obtaining DVD information.

Methods:

- int getRegionCode(). Returns the region code of the DVD.
- String[] getAudioTracks(). Returns an array with the names of the audio tracks on the DVD.
- String[] getSubtitles(). Returns an array with the languages of the subtitles on the DVD.

Class VHSMovie

The class VHSMovie extends class Movie and implements the interface VHS.

Instance variables:

- String format. The format of the VHS movie
- String language. The language of the VHS movie

Constructor and methods:

- VHSMovie(String initialTitle, String[] initialActors, String initialDirector, String initialFormat, String initialLanguage)
  Creates a VHSMovie object and initializes the instance variables.
- String getFormat(). Returns the value of the variable format.
- String getLanguage(). Returns the value of the variable language.
- String toString(). Returns a string representation of the object with the following format: title, format, language where title is the title of the VHS movie; format is the format of the VHS movie and language is the language of the VHS movie. The fields are delimited by a comma (,). You can assume that the fields themselves do not contain any commas.

Class DVDMovie

The class DVDMovie extends class Movie and implements the interface DVD.

Instance variables:

- int regionCode. The region code of the DVD movie
- String[] audioTracks. The names of the audio tracks on the DVD movie
- String[] subtitles. The languages of the subtitles on the DVD movie

Constructor and methods:

- DVDMovie(String initialTitle, String[] initialActors, String initialDirector, int initialRegionCode, String[] initialAudioTracks, String[] initialSubtitles)
  Creates a DVDMovie object and initializes the instance variables.

- int getRegionCode(). Returns the value of the variable regionCode.
- String[] getAudioTracks(). Returns a reference to the array audioTracks.
- String[] getSubtitles(). Returns a reference to the array subtitles.
- String toString(). Returns a string representation of the object with the following format: title, regionCode where: title is the title of the DVD movie; regionCode is the region code of the DVD movie. The fields are delimited by a comma (,). You can assume that the fields themselves do not contain any commas.

Test driver classes

Complete implementation of the following test drivers is provided in the student archive:

- Class TestMovie
- Class TestVHS
- Class TestDVD
- Class TestVHSMovie
- Class TestDVDMovie

Files

The following files are needed to complete this assignment.

student-files.zip — Download this file. This archive contains the following test drivers:

- TestMovie.java
- TestVHS.java
- TestDVD.java
- TestVHSMovie.java
- TestVHSMovie.java

Tasks

Implement the abstract class Movie, the interfaces VHS and DVD, and the concrete classes VHSMovie and DVDMovie. Document using Javadoc and follow Sun’ s code conventions. The following steps will guide you through this assignment. Work incrementally and test each increment. Save often.

1. Implement the class Movie from scratch. Use TestMovie to test your implementation.
2. Implement the interface VHS from scratch. Use TestVHS to test your implementation.
3. Implement the interface DVD from scratch. Use TestDVD to test your implementation.
4. Implement the class VHSMovie from scratch. Use TestVHSMovie to test your implementation.
5. Implement the class DVDMovie from scratch. Use TestDVDMovie to test your implementation.