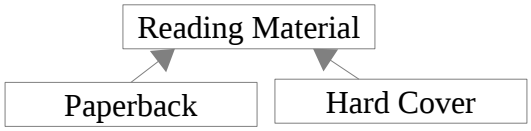
Solutions

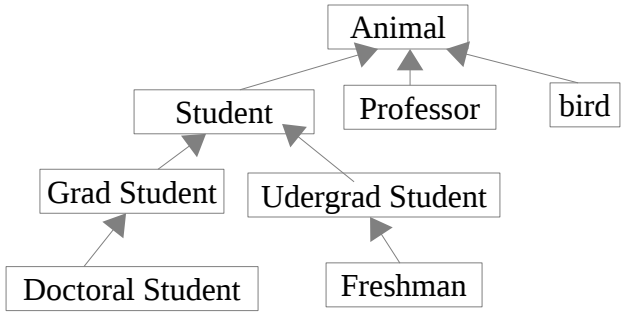
1. Suppose that a Java class *A* extends another class *B*, and implements interface *If1*. *B* extends the abstract class *Ab* and implements interface *If2*.
(a) What are the requirements that *A* should meet, in terms of contracts with other classes and interfaces?
Answer:
A should implement all methods in If1.

(b) What are the requirements that *B* should meet, in terms of contracts with other classes and interfaces?
Answer:
B should implement all methods in If2, and any methods declared abstract in Ab.

2. For each of the following sets, construct a possible inheritance hierarchy of classes that model relationships of objects in the set - use your intuition and common sense; just draw pictures, nothing else. Some objects may not fit into any hierarchy that you construct – in that case, draw them as separate classes.

(a) {"Reading Material", Alphabet, Words, Paperback, "Hard Cover"}
(b) {Student, "Undergrad Student", "Grad Student", Freshman, "Doctoral Student", Animal, Bird, Professor, "Doctoral Degree"}
Answer:





Other objects do not fit into any hierarchies other than themselves. The relationship between Alphabet and Word is a "uses" relation, not a "is-a" relation.

3. Suppose you have two Java packages edu.newpaltz.pds and edu.newpaltz.test. A and B are in package edu.newpaltz.pds, B extends A. C and D are in edu.newpaltz.test and C extends A. If A has a method f() with default (package) access and a method g() with protected access, describe objects (of which classes) have access to f() and g().

Answer:
Since f() has default (package) access, only objects of classes in the same package have access to it.
Objects of type _____A and B_____have access to f().

Since g() has protected access, Objects of the same package or Objects of subclasses can access it.
Objects of type _____A, B and C_____have access to g().

4. Consider:

```
interface I1 {
   public void f();
}

class A implements I1 {
   public void f() {
         //implementation
   }
   public static void g(){
         //implementation
   }
}

class B extends A {
   public void f() {
         //implementation
   }
   public static void g(){
         //implementation
   }
}
```

Suppose

```
I1 i1=new B();
I1 i2=new A();
A a1= new A();
A a2=new B();
A b1= (A) new B();
```

Without writing any code, explain in words the effect of each of the following method calls. Just say things like "This will call f method implemented in A" etc.

```
b1.f();
b1.g();
i1.f();
a1.g();
i2.f();
a2.g();
```

Solution

method	***will call***
b1.f();	f in B
b1.g();	g in A
i1.f();	f in B
a1.g();	g in A
i2.f();	f in A
a2.g();	g in A

Test Code

```
interface I1 {
   public void f();
}

class A implements I1 {
   public void f() {
         System.out.println("f in A");
   }
   public static void g(){
         System.out.println("g in A");
   }
}

class B extends A {
   public void f() {
         System.out.println("f in B");
   }
   public static void g(){
         System.out.println("g in B");
   }
}

class tester {
      public static void main(String[] args){

         I1 i1=new B();
         I1 i2=new A();
         A a1= new A();
         A a2=new B();
         A b1= (A) new B();
         b1.f();
         b1.g();
         i1.f();
         a1.g();
         i2.f();
         a2.g();
   }
}
```

5. Visibility Exercise

Consider classes A through G:

```
package my.p1;
public class A {
public int x;
int y;
protected z;
private t;
.....
}
class B {
.....
}
class C extends A {
....
}
```

```
package my.p2;
import my.p1.A;
public class D extends A{

.....
}
class E {
.....
}
```

```
package my.p2.q1;
import my.p1.A;
public class F extends A{

.....
}
class G {
.....
}
```

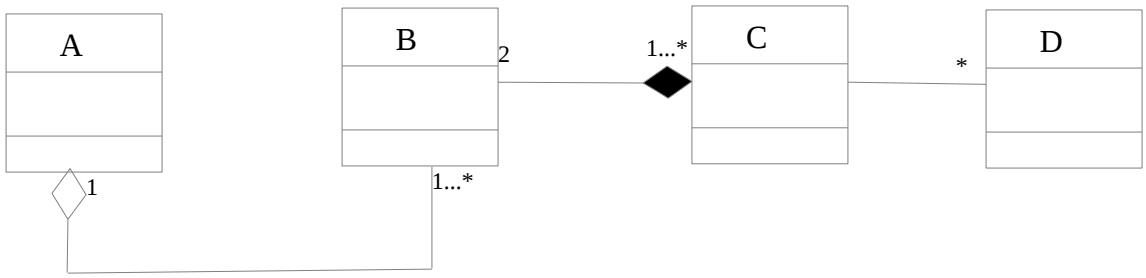
Based on the structure of classes shown, fill in the following visibility diagram showing the visibility of members of class A to the other classes:

Modifier (**of a member of A**)	**A**	**B**	**C**	**D**	**E**	**F**	**G**
public	Y	Y	Y	Y	Y	Y	Y
private	Y	N	N	N	N	N	N
protected	Y	Y	Y	Y	N	Y	N
no modifier	Y	Y	Y	N	N	N	N

6. Draw a sequence diagram for checking out a movie from the Red Box console at your local grocery store. Assume the following sequence of actions.

The main screen has options Rent and Return.
From the rent menu, one could browse the movies, select, and them to the cart.
To check out one should swipe the credit card and for security input the billing address zip code.
Charge the credit card
Print a receipt
Deliver the movie to the customer

7. In the UML class diagram below, describe (in words) the relationship between various classes in as much detail as possible, including the multiplicities of associations in each case.



A *aggregates* 1 or more B objects. For example,

```
class A {
        B b1, b2;
        .....}
```

A includes B via references, so that the referenced B objects exist independent of A objects

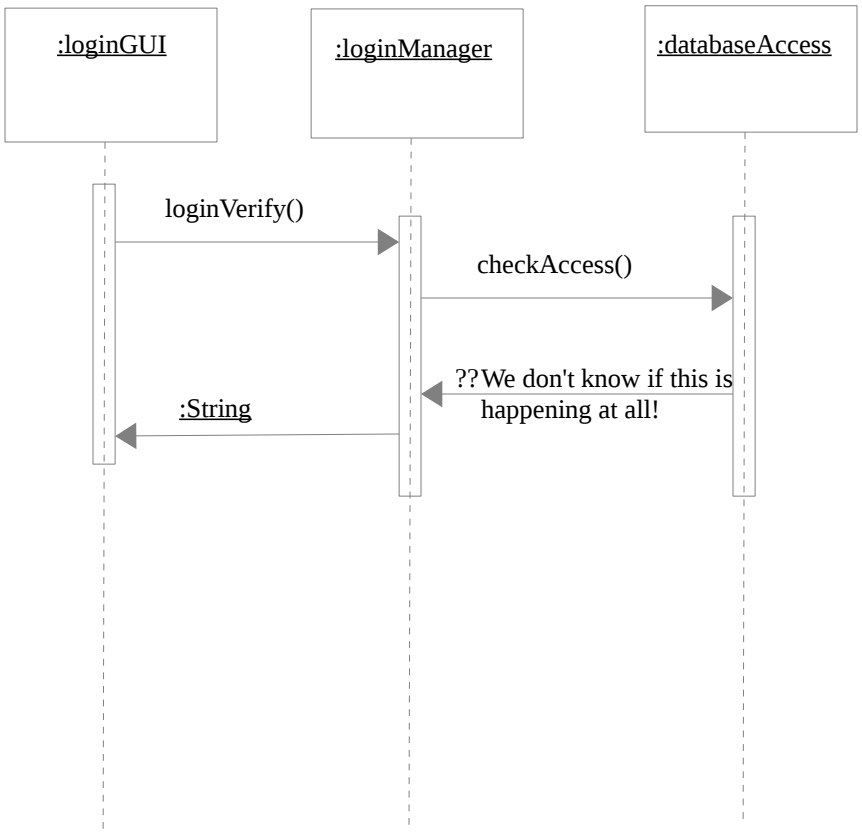
C is *composed* of 2 B Objects. For example,

```
class C {
        B b1= new B();
        B b2= new B();
        .......}
```

C includes B via construction, so that the referenced B objects do not exist independent of C objects

C is *associated* with 0 or more D objects

8. Suppose you have an elementary Java application, with a “loginGUI” class that presents the user with a login screen. The user inputs username and password, and on clicking “login” button, the “loginGUI” object calls a method , loginVerify(), in a “loginManager” object, with the input values as parameters. The “loginManager” checks username and password with a “dataBaseAccess” object with a checkAccess() method, and returns a string message to the “loginGUI” object. (Don't worry about the nature and handling of this string)
Write a sequence diagram for method calls between the three java classes “loginGUI” and “loginManager”and “dataBaseAccess”.


:loginGUI
:loginManager
:databaseAccess
loginVerify()
checkAccess()
??We don't know if this is happening at all!
:String