Exercise 8.2

After creating a comment on the CD object it is also displayed when listing the database, even though the CD was added to the database before setting the comment.

This is because the database has a reference to the CD object and uses this reference to get the informatio about the object each time it prints the list. Only one CD object has been created so the Database must be storing the one that has the comment.

Exercise 8.4

When the 'extends Item' is removed from the source of the Video class, then the inheritance arrow in the diagram disappears.

Exercise 8.5

Yes, it is possible to call inherited methods through the sub-menu called 'inherited from Item'

Exercise 8.6

Add the getTitle method to Item:

```
public String getTitle()
{
    return title;
}
```

Define printShortDetails in CD:

```
public void printShortDetails()
{
    System.out.println(getTitle() + " by " + artist);
}
```

Exercise 8.7

First, the Step Into button takes us to the superclass constructor, which on the subsequent Step Into initializes the fields: title, playingTime, gotIt and comment. Then it returns back to the CD constructor an initializes the last two fields: artist and numberOfTracks.

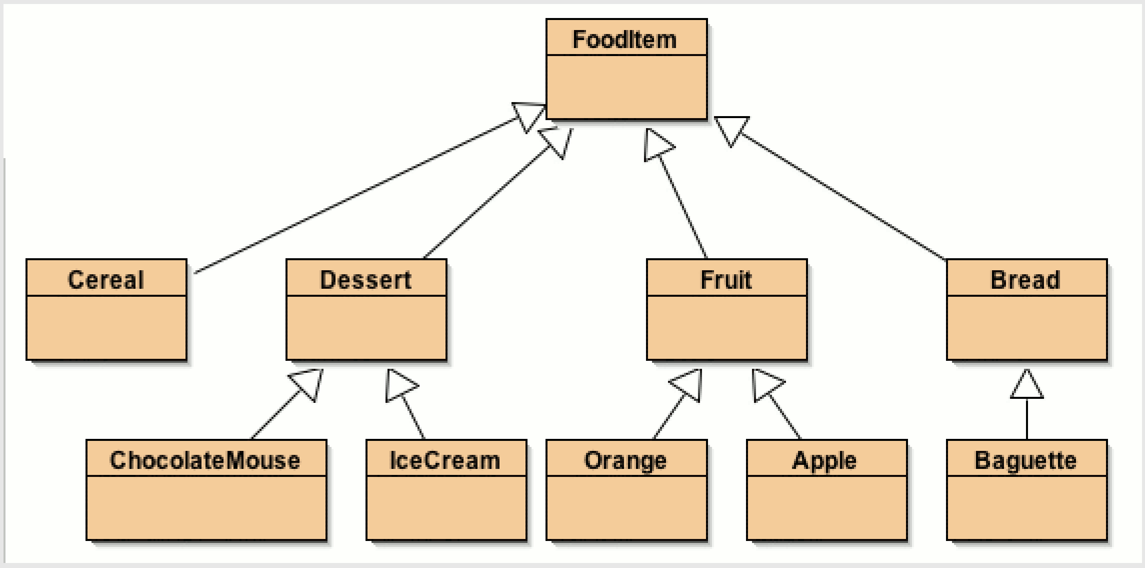
Exercise 8.8

The VideoGame:

```
/**
 * The VideoGame class represents a video game object. Information about the
 * video game is stored and can be retrieved.
 *
 * @author Poul Henriksen
 * @version 2005-10-03
 */
public class VideoGame extends Item
{
    private String platform;
    private int maxPlayers;

    /**
     * Constructor for objects of class Video
     */
    public VideoGame(String theTitle, String platform, int maxPlayers, int time)
    {
        super(theTitle, time);
        this.platform = platform;
        this.maxPlayers = maxPlayers;
    }

    /**
     * Return the platform this game runs on.
     */
    public String getPlatform()
    {
        return platform;
    }

    /**
     * Return the maximum number of players this game is for.
     */
    public int getMaxPlayers()
    {
        return maxPlayers;
    }
}
```

Exercise 8.9.

One possible hierarchy:



Exercise 8.10

A touch pad and mouse are both input devices for a computer. They are very similar and they could eithe
have a common superclass (InputDevice) or one could be a superclass of the other.

Exercise 8.11

Argument for a square being a subclass of a rectangle:

- a square is just a rectangle where the sides are restricted to be of equal length.

Argument for a rectangle being a subclass of a square:

- a rectangle is a square that just has an extra attribute: the ratio between the sizes of the width and height

Argument for neither:

- a rectangle has two attributes determining its shape and a square has just one.

If the two attributes of a Rectangle have the same value, is it equivalent to a Square object?

Exercise 8.12

Which of the following assignments are legal, and why?

a) Person p1 = new Student();

- This is legal because Student is a subclass of Person.

b) Person p2 = new PhDStudent();

- This is legal because PhDStudent is a subclass of Person (because it is a subclass of Student which is a subclass of Person)

c) PhDStudent phd1 = new Student();

- This is not legal, because a student is not a subclass of PhDStudent.

d) Teacher t1 = new Person();

- This is not legal because a Person is not a subclass of Teacher.

e) Student s1 = new PhDStudent();

- This is legal, because PhDStudent is a subclass of Student.

Assume that the two illegal lines above are changed to:

```
PhDStudent phd1;// = new Student();
Teacher t1;// = new Person();
```

f) s1 = p1

- This is not legal, because a Person is not a subclass of Student. The compiler only knows the static type of p1 which is Person - it does not know the dynamic type which is Student.

g) s1 = p2

- This is not legal, because a Person is not a subclass of Student (same arguments as in f).

h) p1 = s1

- This is legal because Student is a subclass of Person.

i) t1 = s1

- This is not legal because Student is not a subclass of Teacher.

j) s1 = phd1

- This is legal because PhDStudent is a subclass of student.

k) phd1 = s1

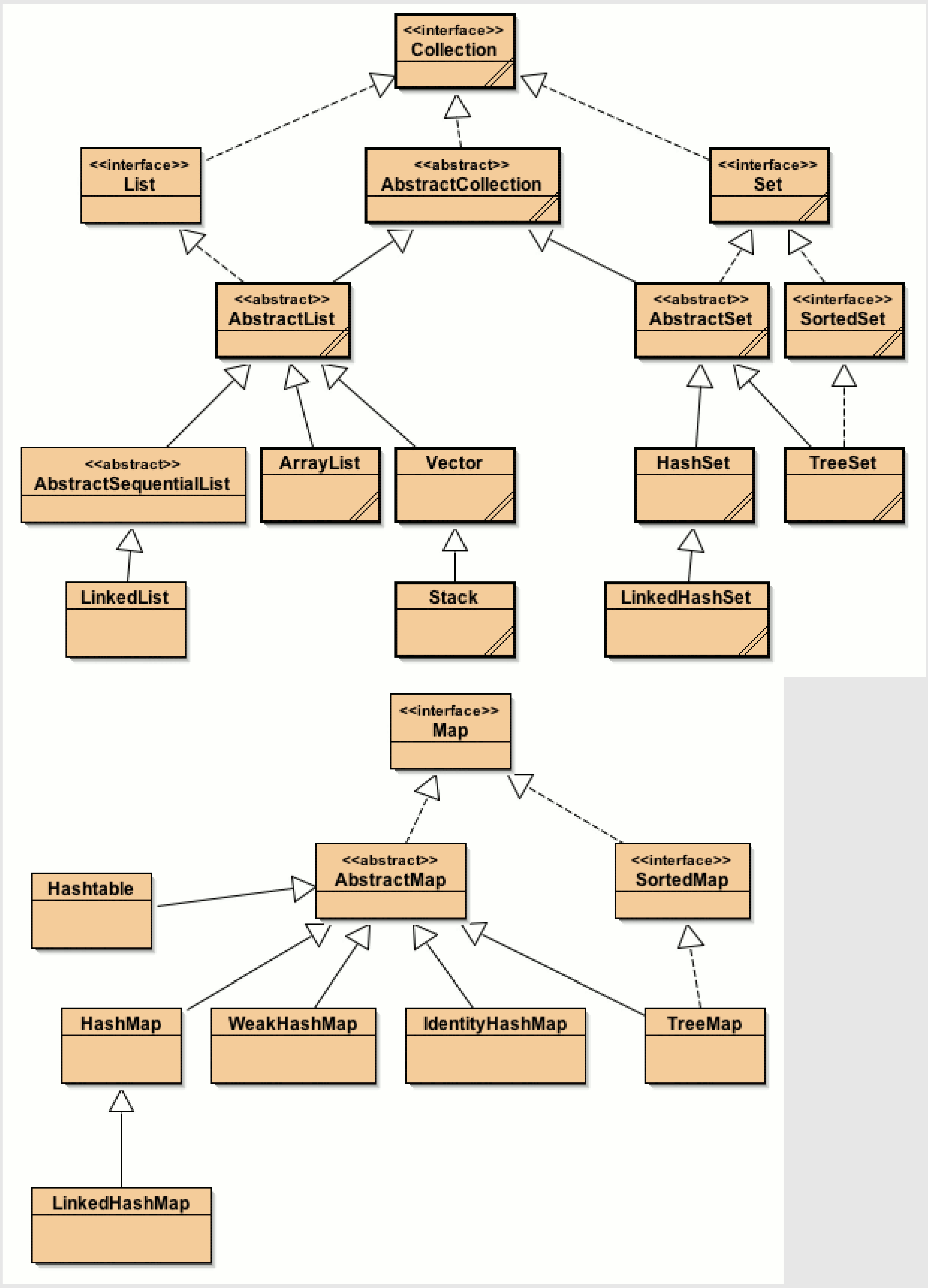
- This is not legal because Student is not a subclass of PhDStudent.

Exercise 8.14

Nothing has to change in the Database class when we add a new Item subclass. This is because the Database never worries about the actual subclass, but treats all objects as Items.

Exercise 8.15

From the documentation to JDK 1.5.0 the following class hierarchies can be drawn:

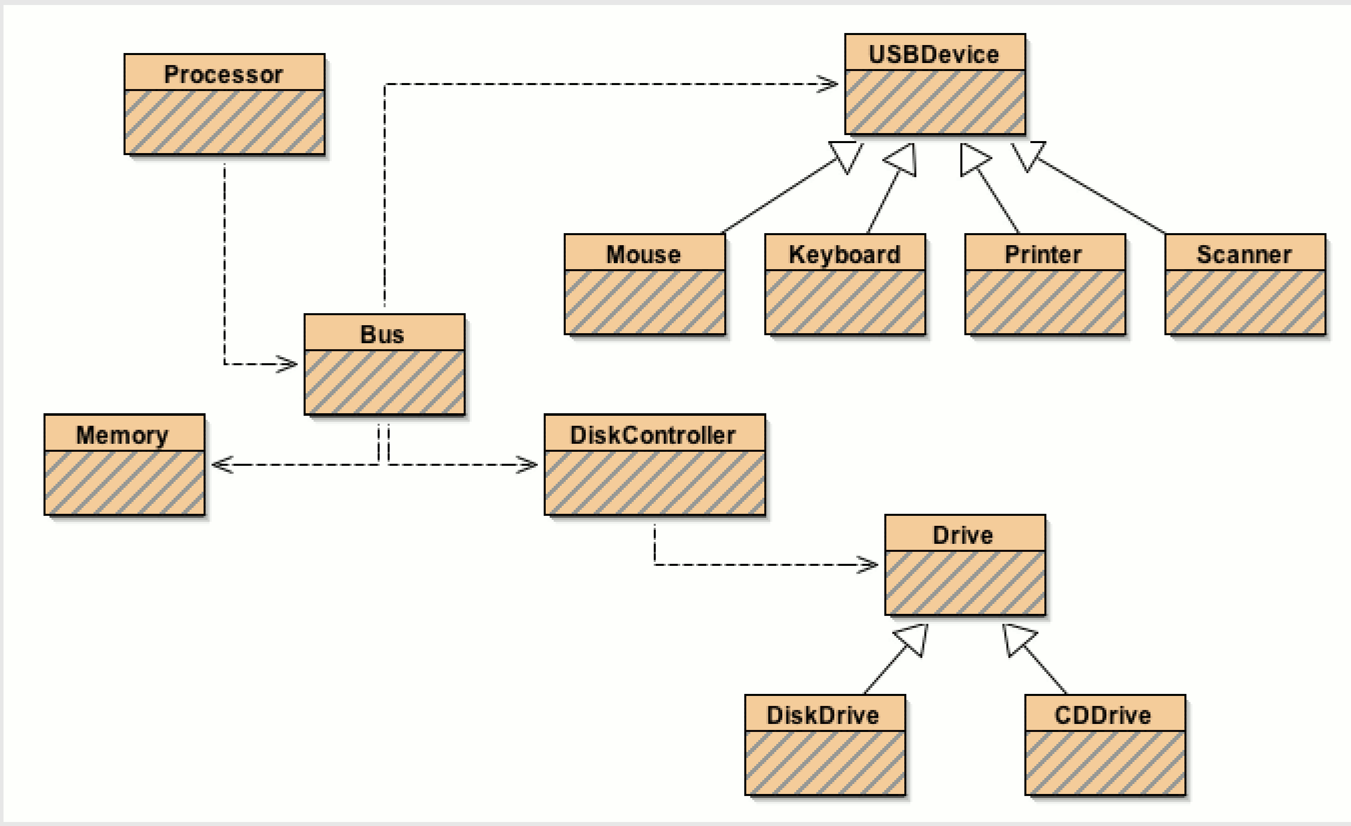


Exercise 8.16

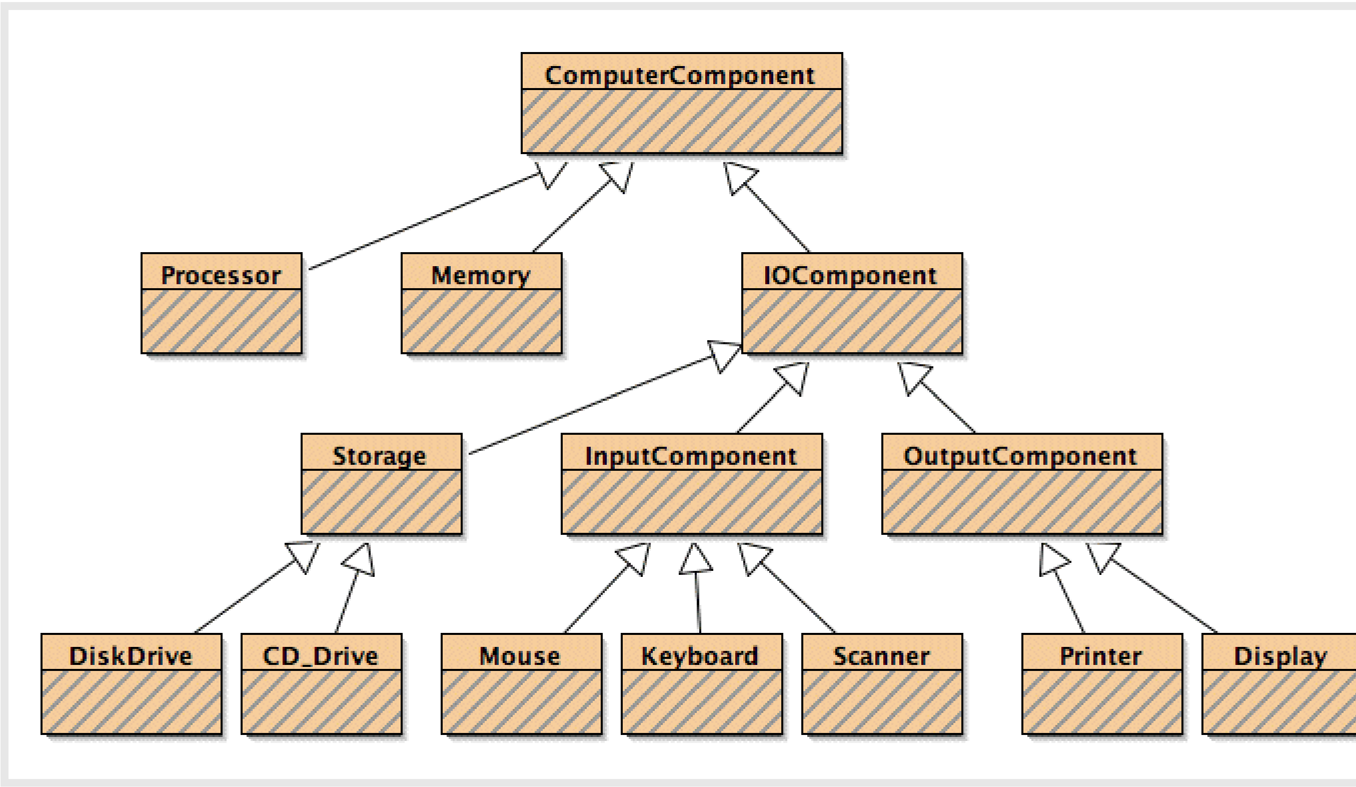
Download: 08-16-lab-classes.zip

Exercise 8.17

An example of a class hierarchy of some of the components in a computer:



Or maybe this:



Exercise 8.18

The legal statements tells us the following:

a) m=t
This tells us that T is subclass of M

b) m=x
This tells us that X is a subclass of M

c) o=t
This tells us that T is a subclass of O. But T was also a subclass of M, and Java does not allow multiple inheritance. Therefore M must be a subclass of O or vice versa.

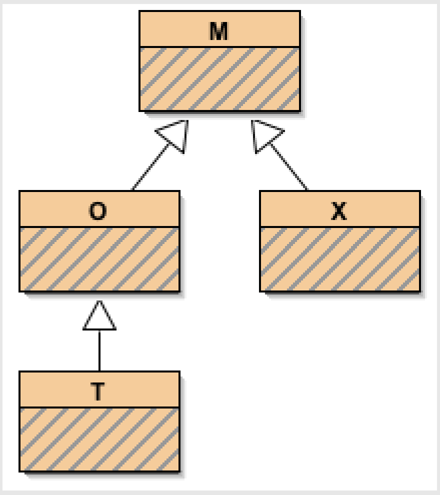
And the illegal statements gives us:

d) o=m
M is **not** a subclass of O. Hence from c) we have that O is a subclass of M.

e) o=x
X is **not** a subclass of O

f) x=o
O is **not** a subclass of X

From this information we can be sure that the following relations exist:



Exercise 8.19

See the diagram in the solution to exercise 8.15.