

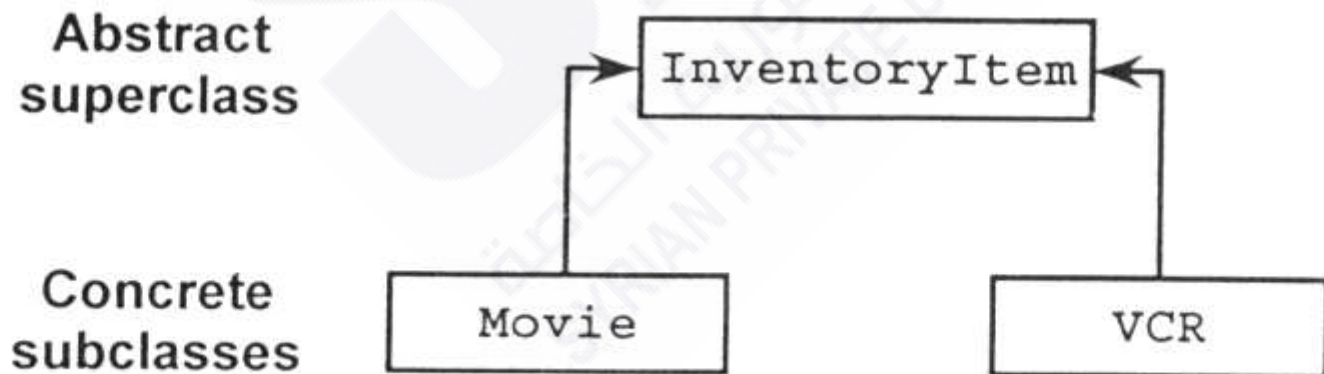
Structuring Code by Using Abstract Classes and Interfaces



Defining Abstract Classes

- An abstract class cannot be instantiated.
- Abstract methods must be implemented by subclasses.
- Interfaces support multiple inheritance.

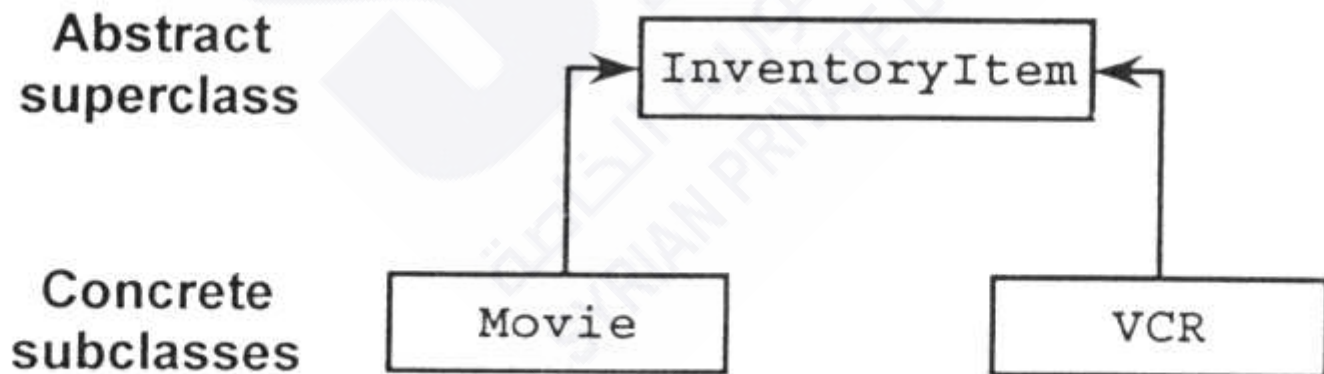
is the specification of a set of methods,
which is similar to an abstract class.



Defining Abstract Classes

- An abstract class cannot be instantiated.
- Abstract methods must be implemented by subclasses.
- Interfaces support multiple inheritance.

Provide multiple inheritance. Class can implement an unlimited number of interfaces but can only extend one superclass.



Creating Abstract Classes

- Use the `abstract` keyword to declare a class as abstract.

```
public abstract class InventoryItem {  
    private float price;  
    public boolean isRentable()...  
}
```

The diagram illustrates inheritance. A central box contains the code for the `InventoryItem` abstract class. Two arrows point from this box to two separate boxes below it. The left box contains the code for the `Movie` class, which `extends InventoryItem`. The right box contains the code for the `Vcr` class, which also `extends InventoryItem`. This visualizes how both `Movie` and `Vcr` inherit from the abstract `InventoryItem` class.

```
public class Movie  
extends InventoryItem {  
    private String title;  
    public int getLength()...
```

```
public class Vcr  
extends InventoryItem {  
    private int serialNbr;  
    public void setTimer()...
```

What Are Abstract Methods?

- An abstract method:
 - Is an implementation placeholder
 - Is part of an abstract class
 - Must be overridden by a concrete subclass
- Each concrete subclass can implement the method differently.

Defining Abstract Methods

- Use the `abstract` keyword to declare a method as abstract:
 - Provide the method signature only.
 - The class must also be abstract.
- Why is this useful?
 - Declare the structure of a given class without providing complete implementation of every method.

```
public abstract class InventoryItem {  
    public abstract boolean isRentable();  
    ...  
}
```

Abstract classes can contain method that are not declared as abstract.

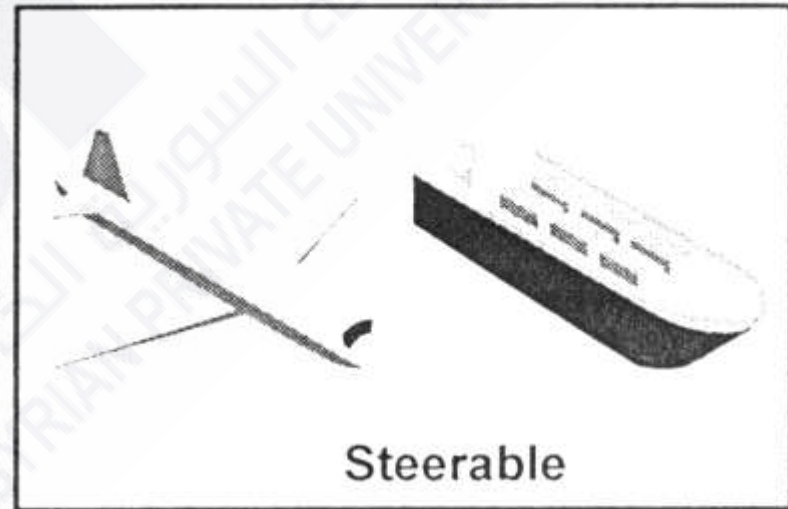
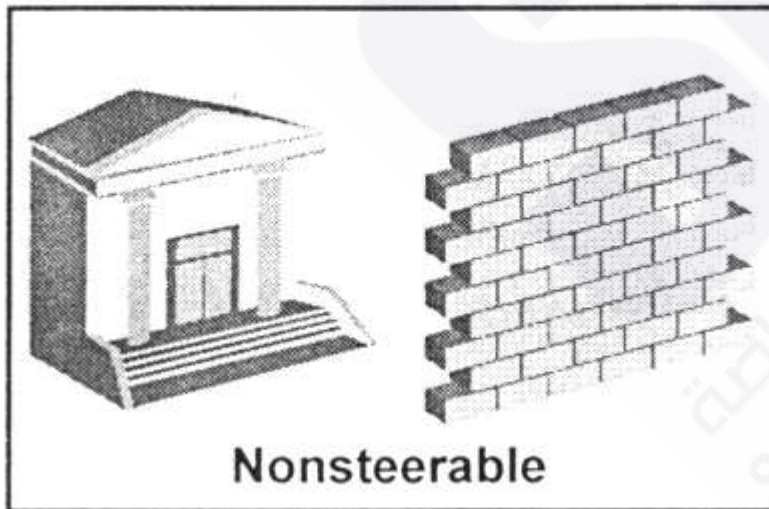
Defining and Using Interfaces

- An interface is like a fully abstract class:
 - All of its methods are abstract.
 - All variables are `public static final`.
- An interface lists a set of method signatures without any code details.
- A class that implements the interface must provide code details for all the methods of the interface.
- A class can implement many interfaces but can extend only one class.

```
class Movie extends InventoryItem implements Sortable , Listable {  
    .....  
}
```

Examples of Interfaces

- Interfaces describe an aspect of behavior that different classes require.
- For example, classes that can be steered support the “steerable” interface.
- Classes can be unrelated.



Creating Interfaces

- Use interface keyword:

```
public interface Steerable {  
    public static final int MAXTURN = 45;  
    public abstract void turnLeft(int deg);  
    void turnRight(int deg);  
}
```

- All methods are public abstract.
- All variables are public static final.

Implementing Interfaces

Use `implements` keyword.

```
public class Yacht extends Boat  
    implements Steerable  
    public void turnLeft(int deg) {...}  
    public void turnRight(int deg) {...}  
}
```