Topic 5: Class Hierarchies

Part b: Polymorphism and Abstract Classes (Ch 4.1 – 4.2)

Polymorphism

Polymorphism

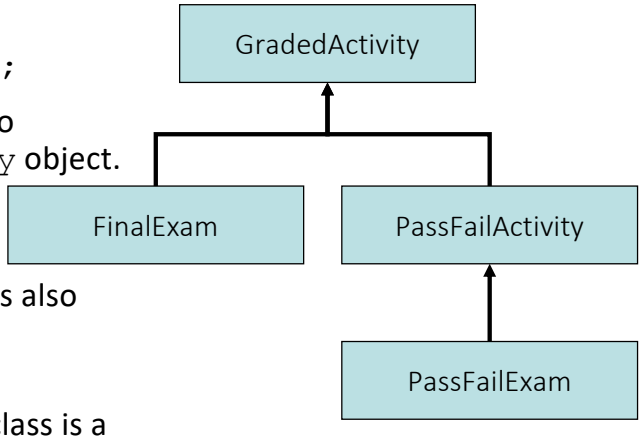
- A reference variable can reference objects of classes that are derived from the variable's class.

```
GradedActivity exam;
```

- We can use the exam variable to reference a `GradedActivity` object.

```
exam = new
GradedActivity();
```

- The `GradedActivity` class is also used as the superclass for the `FinalExam` class.
- An object of the `FinalExam` class is a `GradedActivity` object.



Polymorphism (2)

- A `GradedActivity` variable can be used to reference a `FinalExam` object.

```
GradedActivity exam = new FinalExam(50, 7);
```

- This statement creates a `FinalExam` object and stores the object's address in the exam variable.
- This is an example of **polymorphism**.
- The term polymorphism means the ability to take many forms.
- In Java, a reference variable is polymorphic because it can reference objects of types different from its own, as long as those types are subclasses of its type.

Polymorphism (3)

- Other legal polymorphic references:

```
GradedActivity exam1 = new FinalExam(50, 7);
GradedActivity exam2 = new PassFailActivity(70);
GradedActivity exam3 = new PassFailExam(100, 10, 70);
```

- The `GradedActivity` class has three methods: `setScore`, `getScore`, and `getGrade`.
- A `GradedActivity` variable can be used to call only those three methods.

```
GradedActivity exam = new PassFailExam(100, 10, 70);
System.out.println(exam.getScore());
System.out.println(exam.getGrade());
System.out.println(exam.getPointsEach());
```

Polymorphism and Dynamic Binding

- If the object of the subclass has overridden a method in the superclass:
  - If the variable makes a call to that method the subclass's version of the method will be run.

```
GradedActivity exam = new PassFailActivity(60);
exam.setScore(70);
System.out.println(exam.getGrade());
```

- Java performs *dynamic binding* or *late binding* when a variable contains a polymorphic reference.
- The Java Virtual Machine determines at runtime which method to call, depending on the type of object that the variable references.

Polymorphism (4)

- It is the object's type, rather than the reference type, that determines which method is called.
- Example:
  - Polymorphic.java
- You cannot assign a superclass object to a subclass reference variable.

Abstract

Classes and Methods

Abstract Classes

- An **abstract class** cannot be instantiated, but other classes are derived from it.
- An Abstract class serves as a superclass for other classes.
- The abstract class represents the generic or abstract form of all the classes that are derived from it.
- A class becomes abstract when you place the `abstract` key-word in the class definition.

```
public abstract class ClassName
```

Abstract Methods

- An **abstract method** has no body and must be overridden in a subclass.
- An ***abstract method*** is a method that appears in a superclass, but expects to be overridden in a subclass.
- An abstract method has only a header and no body.

```
AccessSpecifier abstract ReturnType
                    MethodName(ParameterList);
```

- Example:
  - Student.java, CmptStudent.java, CmptStudentDemo.java

Abstract Methods (2)

- Notice that the key word `abstract` appears in the header, and that the header ends with a semicolon.

```
public abstract void setValue(int value);
```

- Any class that contains an abstract method is automatically abstract.
- If a subclass fails to override an abstract method, a compiler error will result.
- Abstract methods are used to ensure that a subclass implements the method.

Interface

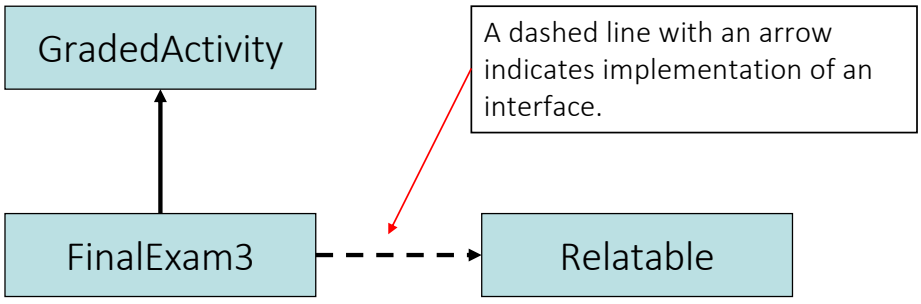
Revisited

Implementing Multiple Interfaces

- A class can be derived from only one superclass.
- Java allows a class to implement multiple interfaces.
- When a class implements multiple interfaces, it must provide the methods specified by all of them.
- To specify multiple interfaces in a class definition, simply list the names of the interfaces, separated by commas, after the implements key word.

```
public class MyClass implements Interface1,
                                 Interface2,
                                 Interface3
```

Interfaces in UML



Polymorphism with Interfaces

- Java allows you to create reference variables of an interface type.
- An interface reference variable can reference any object that implements that interface, regardless of its class type.
- This is another example of polymorphism.
- Example:
  - RetailItem.java
  - CompactDisc.java
  - DvdMovie.java
  - PolymorphicInterfaceDemo.java
- Beginning in Java 8, interfaces may have *default methods*.
  - A default method is an interface method that has a body.
  - You can add new methods to an existing interface without causing errors in the classes that already implement the interface.