CSE1720

Week 08, Lecture 12

Winter 2014 ◆ Tuesday, Feb 25, 2014

YORK UNIVERSITÉ UNIVERSITY

General Info

- Welcome back from reading week!
- Assigned reading was Chapter 9
- Here's our plan
  - Complete (conceptual) exercises related to Ch 9 material (Tue Feb 25, Thu Feb 27)
  - Complete (conceptual) exercises related to Ch 10 material (Tue Mar 04, Thu Mar 06)
  - Complete lab exercises related to building collections into our game software, add in interactivity
- Labtest on Thu Mar 13/Fri Mar 14: ActionListener, Collections
- Written test on Mar 20th will cover Ch9 and Ch10.

YORK UNIVERSITÉ UNIVERSITY

Game software: Recap

- One observer design pattern is implemented
  - a `Timer` instance is our observee; it dispatches action events to trigger frame refreshes
  - a `FrameAdvancer` instance is our observer; it receives these events and redraws the frame
- Encapsulation/Abstraction:
  - the "logic" that specifies what actions are happening in the "game world" are located within the `FrameAdvancer` class definition
  - the logic that specifies how the entities in the game world get shown on the frame are located within the `FrameAdvancer` class definition
  - we will work on abstracting these components away…

3

YORK UNIVERSITÉ UNIVERSITY

Overview

- Chapter 9 topic: Inheritance
- We have been making use of inheritance concepts all throughout the course
- Now we will revisit the concepts

4

YORK UNIVERSITÉ UNIVERSITY

Questions about Inheritance

- General properties of subclasses: **RQ 9.1 – 9.7, 9.29**
- Rationale for subclass design: **RQ 9.8 – 9.9**
- Substitutability principle: **RQ 9.10 – 9.12**
- Early vs Late Binding: **RQ 9.13 – 9.19**
- Manual Casting (in Subclass Context): **RQ 9.20 – 9.23**
- Interfaces, Abstract Classes: **RQ 9.24, 9.27 – 9.28**
- Instantiation (in Subclass Context): **RQ 9.25 – 9.26**
- Strong typing: **RQ 9.30 – 9.32**

YORK UNIVERSITÉ UNIVERSITY

5

Review

Chapter 1, In More Depth 1.6, p. 24

What is a “type”?

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

6

Non-primitive types

- A type is as a type does
- A **class definition** defines a type
- An **interface** defines a type
- An **abstract class definiton** defines a type

- In strongly typed programming languages, variables must be declared to be of a particular type in order to be used
  - this presents some limitations, but provides many benefits

YORK UNIVERSITÉ UNIVERSITY

Examples of flexibility at runtime

Consider the following:

```
for (Shape s : myCollection) {
      s.draw(graphics);
}
```

- the `draw` method is defined for *all* Shape objects.
- at runtime, the VM doesn't need to know which specific Shape objects are in the collection
- the design of the various different types of shape makes use of clever delegation.
- Each one of the different types has its own way of deciding how it should be drawn, when given a `Graphics2D` object

YORK UNIVERSITÉ UNIVERSITY

The "Tension"

- At compilation:
  - the compiler looks to the variable's type to determine whether a statement is syntactically and semantically correct
- At runtime:
  - the VM uses the services defined by the object's type to perform the specified operations.
- We want the benefits of the compiler's type checking, AND we want the benefits of flexible run-time behaviour
  - polymorphism is the solution to this

9

YORK UNIVERSITÉ UNIVERSITY

Questions about Inheritance

- General properties of subclasses: **RQ 9.1 – 9.7, 9.29**
- Rationale for subclass design: **RQ 9.8 – 9.9**
- Substitutability principle: **RQ 9.10 – 9.12**
- Early vs Late Binding: **RQ 9.13 – 9.19**
- Manual Casting (in Subclass Context): **RQ 9.20 – 9.23**
- Interfaces, Abstract Classes: **RQ 9.24, 9.27 – 9.28**
- Instantiation (in Subclass Context): **RQ 9.25 – 9.26**
- Strong typing: **RQ 9.30 – 9.32**

10

YORK UNIVERSITÉ UNIVERSITY

General Properties

RQ9.1 How do you determine whether a class extends another… :

- given the class' API?
- given a UML class diagram?
- given the code (class definition) written in Java?

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

General Properties

Identify parent-child class relationships in our game codebase…

What can you say about the features of the child (cf the parent's features)?

*a class's *features* are its attributes and methods

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

General Properties

RQ9.5 Can a child class have more than one parent class?

Can a parent class have multiple child classes?

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

General Properties

Identify, in the codebase, a place where a child class has a method that has the same name as a parent's method

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

General Properties

RQ9.6 If a subclass has a method with the same name as a parent's method, which method will appear in the subclass API and in which table?

Will your answer change if the child's method has the same parameter list as that of the parent?

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

About the methods in a child class…

They fall under the following **three** categories:

- **new** method
  - if the method signature is defined **ONLY** in the child class and not defined in parent (applies even if method name is found in parent, but signature if different)
- **inherited** method
  - if the method signature defined **ONLY** in the parent class; this method is also available to child instances
- **overridden** method
  - if method signature defined in the parent class **AND also** defined in the child class
  - the child class provides another version of the method functionality that **overrides** the parent's method functionality

YORK UNIVERSITÉ UNIVERSITY

About the fields in a child class…

They fall under the following **three** categories:

- **new** field
  - if the field is defined **ONLY** in the child class and not defined in parent
- **inherited** field
  - if the field defined **ONLY** in the parent class; the child can use it as its own
- **overshadowed** field
  - if a field name in the child is the SAME as a field in the parent (regardless of type). Parent field cannot be used, since the child's field will block it

YORK UNIVERSITÉ UNIVERSITY

General Properties

RQ9.29 If class C extends class P, is C a descendant of the Object class?

What ***kind*** of descendant class?

A direct descendant or indirect descendant?

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

Review

This description of “type” was intended for primitive types.

How do we extend the description to capture non-primitive types?

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

Substitutability

RQ9.10 What is the substitutability principle?

In Java? In everyday life? (RQ9.12)

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

Substitutability

Identify places within the codebase in which the substitutability principle has been used

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

Substitutability

RQ9.11 In which contexts can the substitutability principle be applied?

ANSWER:

YORK UNIVERSITÉ UNIVERSITY

Binding

What is meant by early binding?

What is meant by late binding?

ANSWER:

YORK UNIVERSITÉ UNIVERSITY