CSE 1720

Lecture 6

Aggregation, Graphics IV

Announcements:

- Lectures 7-10 assigned reading: Ch 9, JBA

Goals/To do:

- How to create, copy, and delegate to aggregates
  - example aggregates: `Pixel`, `Picture`, `Graphics2D`
- Create, modify, and iterate over collections
- Implement traversal over a collection
- Implement search within a collection
- Use services of `Graphics2D` for drawing

Goals/To understand:

- recognize aggregates from their APIs
- characterize and distinguish between two traversal techniques
- distinguish between aliases, shallow copies, and deep copies of aggregates
- understand the characteristics of the "current settings" graphical model

Today's Topic

- Issues with making a copy of an aggregate

The Aggregation Relationship

- A class `C` is said to be an **aggregate** if and only if one of its attributes is an object reference (say of type `T`)
- *Aggregation* is the name of the relationship between `C` and `T`.
  - an object of type `C` **HAS-A** object of type `T`
  - the object of type `T` is called the *aggregated part*
- Key observation:
  - it is possible that the object of type `T` can be may have a different lifetime of the object of type `C`
- We will demonstrate this next…
  (but to do this we must first explain the `Stock` and `Investment` classes)

The class `Stock`

- We will use the `Stock` class from type.jar for this example
- A *public* company is a company that offers its stock/shares for sale to the general public, typically through a stock exchange
- A public company has a full name and is represented by a two-character symbol
  - e.g., name: “Alpha Bravo Co.”, symbol: “.AB”
- At any given point in time, the company’s shares have a **selling price**.
- We use the class `Stock` to encapsulate a single share

The class `Stock`

- When constructing a `Stock` instance, the client must specify the two-character symbol.
- The `Stock` class’ `getName()` accesses the name of the company that corresponds to the stock’s two-character stock exchange symbol:

```
ALPHA of BRAVO Company
Alpha of Bravo Company
```

- Whether the name is upper-case or camel-case, this is determined by the boolean flag `titleCaseName`
- The attribute is **public** and **static**
- See `L06App01.java`

The class `Stock`

- The `Stock` class’ `toString()` produces a “nice” string representation consisting of something like:

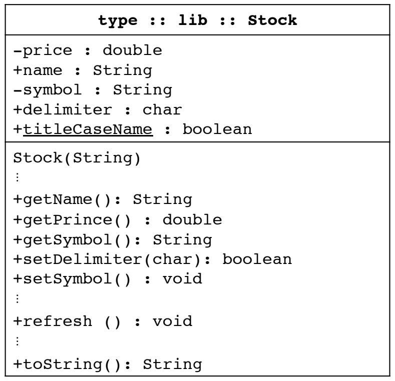
```
.AB*ALPHA of BRAVO Company
.AB:ALPHA of BRAVO Company
.AB+ALPHA of BRAVO Company
.AB ALPHA of BRAVO Company
.AB#ALPHA of BRAVO Company
.AB.ALPHA of BRAVO Company
```

- The character is red is called the delimiter
- The client can specify the character to be used for this delimiter
- See `L06App02.java`

The class `Stock`

- The `Stock` class' `getPrice()` retrieves the most-recently fetched version of the price. Upon instantiation, the current price is fetched.
- The method `refresh()` will connect to the Stock Exchange server and fetch the current version of the price

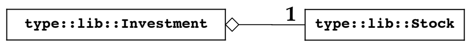
UML Diagram



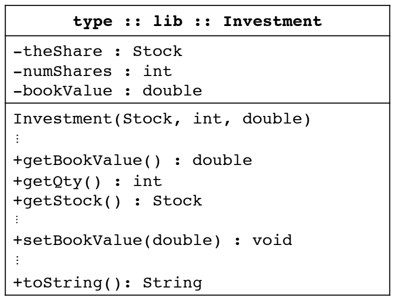
```
        type :: lib :: Stock
-price : double
+name : String
-symbol : String
+delimiter : char
+titleCaseName : boolean
Stock(String)
⋮
+getName(): String
+getPrince() : double
+getSymbol(): String
+setDelimiter(char): boolean
+setSymbol() : void
⋮
+refresh () : void
⋮
+toString(): String
```

The class `Investment`

- The `Investment` class represents the purchase of a certain number of shares at a certain price.
- The book value of an investment is the cost of the shares multiplied by the number of shares
  - If we purchase 10 shares of the Alpha of Bravo Co. at $100, the book value is $1,000
- Investment is an aggregate, its aggregate part is a Stock object



UML Diagram



```
        type :: lib :: Investment
-theShare : Stock
-numShares : int
-bookValue : double
Investment(Stock, int, double)
⋮
+getBookValue() : double
+getQty() : int
+getStock() : Stock
⋮
+setBookValue(double) : void
⋮
+toString(): String
```

The class `Investment`

- If the value of the Alpha of Bravo Co. shares change, then our investment will generate profit/loss
  - E.g., If we purchase 10 shares of the Alpha of Bravo Co. at $100, the book value is $1,000
  - if the company is now valued at $150/share, then our investment has given us a profit of $500
  - $500 = $1500 (current value) - $1000 (book value)
- See `L06App03.java`

Back to our main point…

- it is possible that the object of type `T` can be may have a different lifetime of the object of type `C`
- see `L06App04.java`

This leads to another issue…

- There are two possible copies of an investment:
  - shallow copy
  - deep copy
- With a shallow copy, we will have two different investment objects, but they will both share the same aggregated parts
- see `L06App05.java`

What is a shallow copy?

- object `B` is a *shallow* copy of aggregate `A` when:
  - `A` and `B` are different objects with the same state, and
  - any aggregated parts are shared.
- a change to the aggregated part of object `A` will change the state of object `B`
  - see `L06App05.java`

What is a deep copy?

- object `B` is a *deep* copy of aggregate `A` if:
  - `A` and `B` are different objects with the same state.
  - `A` and `B` are truly different objects, each with its own copies of any aggregated parts
- see `L06App06.java`

Follow-up…

- This material is covered in detail in section 8.1.3
- ensure you read carefully and understand
- your understanding of the concepts of aggregation, shallow copies, and deep copies will be examined on the first written test.