Aggregation (Chapter 8)

CSE 1020

moodle.yorku.ca

Introduction to Aggregation

Most real-life objects are compound. That is, the objects themselves are made up of other objects.

A university consists of various departments and each department has a number of professors.

A creditcard contains the name of the holder and the expiry date.

An investment consists of a stock and each stock has a stock symbol.

Introduction to Aggregation

Combine simple data into more complex data.

1959	COBOL	
1972	C	structures
1979	ML	records
1995	Java	classes

Roots of Aggregation

The notion aggregation can be traced back to the notion of records that could already be found in the programming language COBOL (COmmon Business-Oriented Language) in 1959.

In 1997, 80 percent of the world's businesses ran on COBOL and 310 billion lines of COBOL were in use.

Grace Murray Hopper

COBOL was based on the philosophy of Grace Murray Hopper that programs could be written in a language that was close to English.

The annual "Grace Murray Hopper award for outstanding young computer professionals" was established in 1971 by the Association for Computing Machinery (ACM).

Grace Murray Hopper

(1906–1992)

Aggregate

Definition

A class is called an *aggregate* if it has at least one attribute whose type is not primitive.

Example

The class `Stock` of the package `type.lib` is an aggregate because it has an attribute named `symbol` of type `String`.

The class `Investment` of the package `type.lib` is an aggregate because it has an attribute named `stock` of type `Stock`.

The class `Fraction` of the package `type.lib` is not an aggregate because all its attributes are of primitive type.

Aggregation

Definition

Aggregation is a binary relation on classes. The pair (A, P) of classes is in the aggregation relation if class A (aggregate) has an attribute of type P (part).

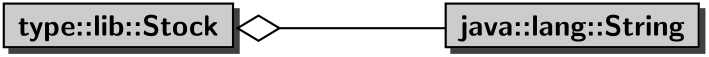
The aggregation relation is also known as the has-a relation. Instead of saying that (A, P) is in the aggregation relation, we often simply say that A has-a P.

Example

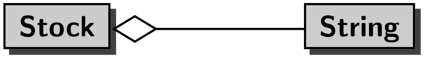
`Stock` has-a `String`.

`Investment` has-a `Stock`.

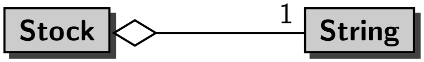
UML Diagrams



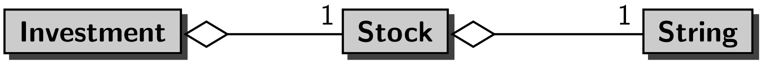
UML Diagrams



UML Diagrams



UML Diagrams



Stock Object

Question

How do you create a `Stock` object with symbol `"HR.A"`?

Stock Object

Question

How do you create a `Stock` object with symbol `"HR.A"`?

Answer

```
String symbol = new String("HR.A"); // "HR.A"
Stock stock = new Stock(symbol);
```

Stock Object

Question

How do you create a `Stock` object with symbol `"HR.A"`?

Answer

```
String symbol = new String("HR.A"); // "HR.A"
Stock stock = new Stock(symbol);
```

Question

Draw the memory diagram depicting memory at the end of the second line. (The class blocks need not be included.)

Stock Object

Answer

100	main invocation
symbol	200
stock	300
200	String object
	"HR.A"
300	Stock object
symbol	200

Investment Object

Question

How do you create an `Investment` object with three shares of HR.A stock, each of value 10.00?

Investment Object

Question

How do you create an Investment object with three shares of HR.A stock, each of value 10.00?

Answer

```
String symbol = new String("HR.A"); // "HR.A"
Stock stock = new Stock(symbol);
int number = 3;
double value = 10.00;
Investment investment = new Investment(stock, number,
   value);
```

Investment Object

Question

How do you create an `Investment` object with three shares of HR.A stock, each of value 10.00?

Answer

```
String symbol = new String("HR.A"); // "HR.A"
Stock stock = new Stock(symbol);
int number = 3;
double value = 10.00;
Investment investment = new Investment(stock, number,
   value);
```

Question

Draw the memory diagram depicting memory at the end of the fifth line.

Investment Object

100	main invocation
symbol	200
stock	300
number	3
value	10.00
investment	400
200	String object
	"HR.A"
300	Stock object
symbol	200
400	Investment object
stock	300
quantity	3
bookValue	10.00

Accessors

Question

Create a random `Investment` object and print its stock symbol.

Accessors

Question

Create a random `Investment` object and print its stock symbol.

Answer

```
Investment investment = Investment.getRandom();
Stock stock = investment.getStock();
String symbol = stock.getSymbol();
output.println(symbol);
```

Accessors

Question

Create a random `Investment` object and print its stock symbol.

Answer

```
Investment investment = Investment.getRandom();
Stock stock = investment.getStock();
String symbol = stock.getSymbol();
output.println(symbol);
```

Question

Draw the memory diagram depicting memory at the end of the first line.

Accessors

Answer

100	main invocation
`investment`	400
`stock`	
`symbol`	
200	String object
	"HR.Z"
300	Stock object
`symbol`	200
400	Investment object
`stock`	300
`quantity`	8
`bookValue`	25.50

Accessors

Question

```
Investment investment = Investment.getRandom();
Stock stock = investment.getStock();
String symbol = stock.getSymbol();
output.println(symbol);
```

Draw the memory diagram depicting memory at the end of the third line.

Accessors

Answer

100	main invocation
`investment`	400
`stock`	300
`symbol`	200
200	`String` object
	"HR.Z"
300	`Stock` object
`symbol`	200
400	`Investment` object
`stock`	300
`quantity`	8
`bookValue`	25.50

Mutators

Question

Create a random `Investment` object and set its stock symbol "HR.B".

Mutators

Question

Create a random `Investment` object and set its stock symbol "HR.B".

Answer

```
Investment investment = Investment.getRandom();
Stock stock = investment.getStock();
stock.setSymbol("HR.B");
```

Mutators

Question

Create a random `Investment` object and set its stock symbol "HR.B".

Answer

```
Investment investment = Investment.getRandom();
Stock stock = investment.getStock();
stock.setSymbol("HR.B");
```

Question

Draw the memory diagram depicting memory at the end of the second line.

Mutators

Answer

100	main invocation
investment	400
stock	300
200	String object
	"HR.Z"
300	Stock object
symbol	200
400	Investment object
stock	300
quantity	8
bookValue	25.50

Mutators

Question

```
Investment investment = Investment.getRandom();
Stock stock = investment.getStock();
stock.setSymbol("HR.B");
```

Draw the memory diagram depicting memory at the end of the third line.

Mutators

100	main invocation
investment	400
stock	300
200	String object
	"HR.Z"
300	Stock object
symbol	500
400	Investment object
stock	300
quantity	8
bookValue	25.50
500	String object
	"HR.B"

How to Copy an Object?

We will show three ways to copy an object:

- create an alias,
- create a shallow copy, and
- create a deep copy.

The created copies are fundamentally different.

How to Create an Alias?

Question

How to create an alias of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

How to Create an Alias?

Question

How to create an alias of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

Answer

```
Investment alias = investment;
```

How to Create an Alias?

Question

How to create an alias of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

Answer

```
Investment alias = investment;
```

Question

Draw the memory diagram depicting memory at the end of the first line.

Alias

100	`main invocation`
`investment`	400
`alias`	
200	`String` object
	"HR.Z"
300	`Stock` object
`symbol`	200
400	`Investment` object
`stock`	300
`quantity`	8
`bookValue`	25.50

Alias

Question

```
Investment investment = Investment.getRandom();
Investment alias = investment;
```

Draw the memory diagram depicting memory at the end of the second line.

Alias

100	main invocation
investment	400
alias	400
200	String object
	"HR.Z"
300	Stock object
symbol	200
400	Investment object
stock	300
quantity	8
bookValue	25.50

How to Create a Shallow Copy?

Question

How to create a shallow copy of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

How to Create a Shallow Copy?

Question

How to create a shallow copy of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

Answer

```
Stock stock = investment.getStock();
int quantity = investment.getQty();
double bookValue = investment.getBookValue();
Investment shallowCopy = new Investment(stock,
quantity, bookValue);
```

How to Create a Shallow Copy?

Question

How to create a shallow copy of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

Answer

```
Stock stock = investment.getStock();
int quantity = investment.getQty();
double bookValue = investment.getBookValue();
Investment shallowCopy = new Investment(stock,
quantity, bookValue);
```

Question

Draw the memory diagram depicting memory at the end of the first line.

Shallow Copy

100	main invocation
investment	400
stock	
quantity	
bookValue	
shallowCopy	
200	String object
	"HR.Z"
300	Stock object
symbol	200
400	Investment object
stock	300
quantity	8
bookValue	25.50

Shallow Copy

Question

```
Investment investment = Investment.getRandom();
Stock stock = investment.getStock();
int quantity = investment.getQty();
double bookValue = investment.getBookValue();
Investment shallowCopy = new Investment(stock,
quantity, bookValue);
```

Draw only those blocks of the memory diagram that change when reaching the end of the fifth line.

Shallow Copy

100	main invocation
investment	400
stock	300
quantity	8
bookValue	25.50
shallowCopy	500
500	Investment object
stock	300
quantity	8
bookValue	25.50

How to Create a Deep Copy?

Question

How to create a deep copy of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

How to Create a Deep Copy?

Question

How to create a deep copy of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

Answer

```
Stock stock = investment.getStock();
String symbol = stock.getSymbol();
int quantity = investment.getQty();
double bookValue = investment.getBookValue();
String symbolCopy = new String(symbol);
Stock stockCopy = new Stock(symbolCopy);
Investment deepCopy = new Investment(stockCopy,
quantity, bookValue);
```

How to Create a Deep Copy?

Question

How to create a deep copy of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

Answer

```
Stock stock = investment.getStock();
String symbol = stock.getSymbol();
int quantity = investment.getQty();
double bookValue = investment.getBookValue();
String symbolCopy = new String(symbol);
Stock stockCopy = new Stock(symbolCopy);
Investment deepCopy = new Investment(stockCopy,
quantity, bookValue);
```

Question

Draw the memory diagram depicting memory at the end of the first line.

Deep Copy

100	main invocation
investment	400
deepCopy	
200	String object
	"HR.Z"
300	Stock object
symbol	200
400	Investment object
stock	300
quantity	8
bookValue	25.50

Deep Copy

Question

```
Investment investment = Investment.getRandom();
Stock stock = investment.getStock();
String symbol = stock.getSymbol();
int quantity = investment.getQty();
double bookValue = investment.getBookValue();
String symbolCopy = new String(symbol);
Stock stockCopy = new Stock(symbolCopy);
Investment deepCopy = new Investment(stockCopy,
quantity, bookValue);
```

Draw only those blocks of the memory diagram that change when reaching the end of the last line.

Deep Copy

100	main invocation
investment	400
deepCopy	500
500	Investment object
stock	600
quantity	8
bookValue	25.50
600	Stock object
symbol	700
700	String object
	"HR.Z"

Deep Copy

Question

Recall that `String` objects are immutable. Is there any point of having two identical `String` objects in memory?

Deep Copy

Question

Recall that `String` objects are immutable. Is there any point of having two identical `String` objects in memory?

Answer

No. It only wastes memory.

Deep Copy

Question

Recall that `String` objects are immutable. Is there any point of having two identical `String` objects in memory?

Answer

No. It only wastes memory.

Question (revisted)

How to create a deep copy of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

Deep Copy

Question

Recall that `String` objects are immutable. Is there any point of having two identical `String` objects in memory?

Answer

No. It only wastes memory.

Question (revisted)

How to create a deep copy of the following `Investment` object?

```
Investment investment = Investment.getRandom();
```

Answer (improved)

```
Investment deepCopy = new Investment(
  new Stock(investment.getStock().getSymbol()),
  investment.getQty(),
  investment.getBookValue());
```

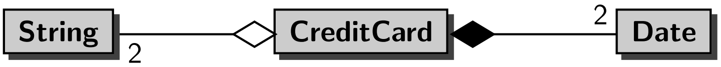
Composition

Composition is a special type of aggregation. The aggregate A and its part P form a composition if "A owns P", that is, each object of type A has exclusive access to its attribute of type P.

The designer and the implementer of a class determine whether an aggregation is a composition.

Java does not provide any special language constructs for implementing compositions. The constructors, accessors and mutators are implemented in a particular way (the details will be covered in CSE1030).

UML Diagrams



Composition

Question

Create a `CreditCard` object with number 123456 and name (of holder) Jane Doe.

Composition

Question

Create a `CreditCard` object with number 123456 and name (of holder) Jane Doe.

Answer

```
CreditCard card = new CreditCard(123456, "Jane Doe");
```

Composition

Question

Create a `CreditCard` object with number 123456 and name (of holder) Jane Doe.

Answer

```
CreditCard card = new CreditCard(123456, "Jane Doe");
```

Question

Draw the memory diagram.

Composition

100	main invocation
card	200
200	CreditCard object
number	300
name	400
issueDate	500
expiryDate	600
300	String object
	"123456"
400	String object
	"Jane Doe"
500	Date object
	now
600	Date object
	two year from now

Accessor

Question

Get the expiry date of the card.

Accessor

Question

Get the expiry date of the card.

Answer

```
CreditCard card = new CreditCard(123456, "Jane Doe");
Date expiryDate = card.getExpiryDate();
```

Accessor

Question

Get the expiry date of the card.

Answer

```
CreditCard card = new CreditCard(123456, "Jane Doe");
Date expiryDate = card.getExpiryDate();
```

Question

Draw the memory diagram depicting memory at the end of the second line.

Accessor

100	main invocation
card	200
expiryDate	700
200	CreditCard object
number	300
name	400
issueDate	500
expiryDate	600
300	String object
	"123456"
400	String object
	"Jane Doe"
500	Date object
	now
600	Date object
	two year from now
700	Date object
	two years from now

Mutator

Question

```
CreditCard card = new CreditCard(123456, "Jane Doe");
Date expiryDate = card.getExpiryDate();
```

Modify the state of the `Date` object named `expiryDate` to 2013.

Mutator

Question

```
CreditCard card = new CreditCard(123456, "Jane Doe");
Date expiryDate = card.getExpiryDate();
```

Modify the state of the `Date` object named `expiryDate` to 2013.

Answer

```
CreditCard card = new CreditCard(123456, "Jane Doe");
Date expiryDate = card.getExpiryDate();
final int YEAR = 113;
expiryDate.setYear(YEAR); // set year to 1900 + YEAR
```

Mutator

Question

```
CreditCard card = new CreditCard(123456, "Jane Doe");
Date expiryDate = card.getExpiryDate();
```

Modify the state of the Date object named expiryDate to 2013.

Answer

```
CreditCard card = new CreditCard(123456, "Jane Doe");
Date expiryDate = card.getExpiryDate();
final int YEAR = 113;
expiryDate.setYear(YEAR); // set year to 1900 + YEAR
```

Question

Draw the memory diagram depicting memory at the end of the code snippet.

Mutator

100	main invocation
card	200
expiryDate	700
YEAR	113
200	CreditCard object
number	300
name	400
issueDate	500
expiryDate	600
300	String object
	"123456"
400	String object
	"Jane Doe"
500	Date object
	now
600	Date object
	two years from now
700	Date object
	one year ago

Composition

Exercise

What output is produced by the following code snippet?

```
CreditCard card = new CreditCard(123456, "Jane Doe");
Date expiryDate = card.getExpiryDate();
output.println(expiryDate);
final int YEAR = 113;
expiryDate.setYear(YEAR);
output.println(expiryDate);
expiryDate = card.getExpiryDate();
output.println(expiryDate);
```

To do

- Study Section 8.1 of the textbook.