Chapter 5: Aggregation and Composition

EECS 1030

moodle.yorku.ca

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 a non-static attribute of type P (part).

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 a non-static attribute of type P (part).

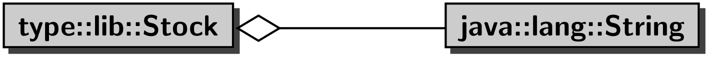
The aggregation relation is also known as the *has-a* relation. Instead of saying that (A, P) is in the aggregation relation, we often simply say that A has-a P.

Aggregation

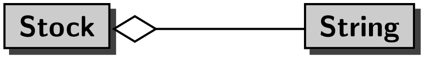
Examples

- A `Stock` has-a `String`
- An `Investment` has-a `Stock`

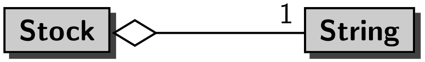
UML Diagrams



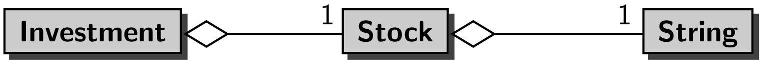
UML Diagrams



UML Diagrams



UML Diagrams



Aggregation

Problem

Implement this API.

Attributes

Question

Besides the constructors, which methods in the API give us a good indication which attributes to introduce?

Attributes

Question

Besides the constructors, which methods in the API give us a good indication which attributes to introduce?

Answer

The accessors `getBookValue`, `getQty` and `getStock`.

Attributes

Question

Besides the constructors, which methods in the API give us a good indication which attributes to introduce?

Answer

The accessors `getBookValue`, `getQty` and `getStock`.

Answer

Which attributes (name and type) should we introduce?

Attributes

Question

Besides the constructors, which methods in the API give us a good indication which attributes to introduce?

Answer

The accessors `getBookValue`, `getQty` and `getStock`.

Answer

Which attributes (name and type) should we introduce?

Answer

```
private double bookValue;
private int qty; // cryptic name
private Stock stock;
```

Constructors, accessors and mutators

Problem

Using eclipse, generate a constructor, and the accessors and mutators.

Constructors, accessors and mutators

Problem

Using eclipse, generate a constructor, and the accessors and mutators.

Questions

Which accessors or mutators should be made private?

Constructors, accessors and mutators

Problem

Using eclipse, generate a constructor, and the accessors and mutators.

Questions

Which accessors or mutators should be made private?

Answer

`setStock`

The `equals` method

Problem

Implement the `equals` method.

Memory diagram

```
Stock stock = new Stock("HR.A");
int quantity = 3;
double bookValue = 2.35;
Investment investment =
  new Investment(stock, quantity, bookValue);
stock.setSymbol("HR.B");
```

Problem

Draw the memory diagram representing memory at the end of line 4.

Memory diagram

```
Stock stock = new Stock("HR.A");
int quantity = 3;
double bookValue = 2.35;
Investment investment =
  new Investment(stock, quantity, bookValue);
stock.setSymbol("HR.B");
```

Problem

Draw the memory diagram representing memory at the end of line 5.

Memory diagram

```
Stock stock = new Stock("HR.A");
int quantity = 3;
double bookValue = 2.35;
Investment investment =
  new Investment(stock, quantity, bookValue);
stock.setSymbol("HR.B");
```

Problem

Draw the memory diagram representing memory at the end of line 5.

Note

The client can directly modify (any part of) the `Investment` object.

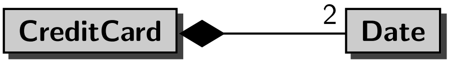
Composition

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.

UML Diagrams



Composition

Problem

Implement this API.

Attributes

Question

Besides the constructors, which methods in the API give us a good indication which attributes to introduce?

Attributes

Question

Besides the constructors, which methods in the API give us a good indication which attributes to introduce?

Answer

The accessors `getBalance`, `getExpiryDate`, `getIssueDate`, `getLimit`, `getName` and `getNumber`.

Attributes

Question

Besides the constructors, which methods in the API give us a good indication which attributes to introduce?

Answer

The accessors `getBalance`, `getExpiryDate`, `getIssueDate`, `getLimit`, `getName` and `getNumber`.

Answer

Which attributes (name and type) should we introduce?

Attributes

Question

Besides the constructors, which methods in the API give us a good indication which attributes to introduce?

Answer

The accessors `getBalance`, `getExpiryDate`, `getIssueDate`, `getLimit`, `getName` and `getNumber`.

Answer

Which attributes (name and type) should we introduce?

Answer

```
private double balance;
private Date expiryDate;
private Date issueDate;
private double limit;
private int number;
```

Constructors, accessors and mutators

Problem

Using eclipse, generate a constructor, and the accessors and mutators. To simplify matters a little, let us exclude the attributes `balance` and `limit`.

Constructors, accessors and mutators

Problem

Using eclipse, generate a constructor, and the accessors and mutators. To simplify matters a little, let us exclude the attributes `balance` and `limit`.

Questions

Which accessors or mutators should be made private?

Constructors, accessors and mutators

Problem

Using eclipse, generate a constructor, and the accessors and mutators. To simplify matters a little, let us exclude the attributes `balance` and `limit`.

Questions

Which accessors or mutators should be made private?

Answer

`setIssueDate`, `setName` and `getNumber`

CreditCard Object

Question

Create a CreditCard object with number 123456 and name Virginia Kaarthouer.

CreditCard Object

Question

Create a CreditCard object with number 123456 and name Virginia Kaarthouer.

Answer

```
int number = 123456;
String name = "Virginia Kaarthouer";
CreditCard card = new CreditCard(number, name);
```

CreditCard Object

Question

Create a CreditCard object with number 123456 and name Virginia Kaarthouer.

Answer

```
int number = 123456;
String name = "Virginia Kaarthouer";
CreditCard card = new CreditCard(number, name);
```

Question

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

CreditCard Object

100	main invocation	
	123456	`number`
	200	`name`
	500	`card`
200	String object	
	"Virginia Kaarthouer"	`value`
300	Date object	
	1415637359054	`time`
400	Date object	
	1478795881318	`time`
500	CreditCard object	
	123456	`number`
	200	`name`
	300	`issueDate`
	400	`expiryDate`

Accessors

Question

Create a `CreditCard` object with number 123456 and name Virginia Kaarthouer and print its expiry date.

Accessors

Question

Create a `CreditCard` object with number 123456 and name Virginia Kaarthouer and print its expiry date.

Answer

```
int number = 123456;
String name = "Virginia Kaarthouer";
CreditCard card = new CreditCard(number, name);
Date expiryDate = card.getExpiryDate();
output.println(expiryDate);
```

Accessors

Question

Create a `CreditCard` object with number 123456 and name Virginia Kaarthouer and print its expiry date.

Answer

```
int number = 123456;
String name = "Virginia Kaarthouer";
CreditCard card = new CreditCard(number, name);
Date expiryDate = card.getExpiryDate();
output.println(expiryDate);
```

Question

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

Accessors

100	main invocation	
	123456	`number`
	200	`name`
	500	`card`
	600	`expiryDate`
200	String object	
	"Virginia Kaarthouer"	`value`
300	Date object	
	1415637359054	`time`
400	Date object	
	1478795881318	`time`
500	CreditCard object	
	123456	`number`
	200	`name`
	300	`issueDate`
	400	`expiryDate`
600	Date object	
	1478795881318	`time`

Accessors

Question

Why can't `card.getExpiryDate()` return a reference to the `Date` object on address 400?

Accessors

Question

Why can't `card.getExpiryDate()` return a reference to the `Date` object on address 400?

Answer

If `card.getExpiryDate()` were to return a reference to the `Date` object on address 400, then both the `main` invocation and the `CreditCard` object would have access to that `Date` object. But the `CreditCard` object "owns" that `Date` object, because `CreditCard` and `Date` form a composition. Hence, `CreditCard` should have exclusive access to that `Date` object.

Accessors

Question

Should we modify the accessor for `expiryDate` generated by eclipse?

Accessors

Question

Should we modify the accessor for `expiryDate` generated by eclipse?

Answer

Yes.

Accessors

Question

Should we modify the accessor for `expiryDate` generated by eclipse?

Answer

Yes.

Problem

Modify the accessors for `expiryDate` and `issueDate`.

Mutators

Question

Create a `CreditCard` object with number 123456 and name Virginia Kaarthouer and set its expiry date to five years from now.

Mutators

Question

Create a `CreditCard` object with number 123456 and name Virginia Kaarthouer and set its expiry date to five years from now.

Answer

```
int number = 123456;
String name = "Virginia Kaarthouer";
CreditCard card = new CreditCard(number, name);
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.YEAR, 5);
Date expiryDate = calendar.getTime();
card.setExpiryDate(expiryDate);
```

Mutators

Question

Create a CreditCard object with number 123456 and name Virginia Kaarthouer and set its expiry date to five years from now.

Answer

```
int number = 123456;
String name = "Virginia Kaarthouer";
CreditCard card = new CreditCard(number, name);
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.YEAR, 5);
Date expiryDate = calendar.getTime();
card.setExpiryDate(expiryDate);
```

Question

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

Mutators

100	main invocation	
	123456	`number`
	200	`name`
	500	`card`
	600	`calendar`
	700	`expiryDate`
200	`String` object	
	"Virginia Kaarthouer"	`value`
300	`Date` object	
	1415637359054	`time`
400	`Date` object	
	1478795881318	`time`
500	`CreditCard` object	
	300	`issueDate`
	400	`expiryDate`
600	`Calendar` object	
	1415637372347	`time`
700	`Date` object	
	1415637372347	`time`

Mutators

Question

Draw the memory diagram depicting memory at the end of the seventh line. Draw only those objects that are relevant to the changes.

Mutators

100	main invocation	
	123456	`number`
	200	`name`
	500	`card`
	600	`calendar`
	700	`expiryDate`
300	Date object	
	1415637359054	`time`
400	Date object	
	1478795881318	`time`
500	`CreditCard` object	
	300	`issueDate`
	800	`expiryDate`
700	Date object	
	1415637372347	`time`
800	Date object	
	1415637372347	`time`

Mutators

Question

Why can't we set the `expiryDate` attribute to refer to the `Date` object on address 700?

Mutators

Question

Why can't we set the `expiryDate` attribute to refer to the `Date` object on address 700?

Answer

If the `expiryDate` attribute were to refer to the `Date` object on address 700, then both the `main` invocation and the `CreditCard` object would have access to that `Date` object. But the `CreditCard` object "owns" that `Date` object, because `CreditCard` and `Date` form a composition. Hence, `CreditCard` should have exclusive access to that `Date` object.

Mutators

Question

Should we modify the mutator for `expiryDate` generated by eclipse?

Mutators

Question

Should we modify the mutator for `expiryDate` generated by eclipse?

Answer

Yes.

Mutators

Question

Should we modify the mutator for `expiryDate` generated by eclipse?

Answer

Yes.

Problem

Modify the mutators for `expiryDate` and `issueDate`.

Constructors

Question

Create a `CreditCard` object with number 123456 and name Virginia Kaarthouer and issue date tomorrow.

Constructors

Question

Create a CreditCard object with number 123456 and name Virginia Kaarthouer and issue date tomorrow.

Answer

```
int number = 123456;
String name = "Virginia Kaarthouer";
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.DAY, 1);
Date issueDate = calendar.getTime();
CreditCard card = new CreditCard(number, name,
issueDate);
```

Constructors

Question

Create a CreditCard object with number 123456 and name Virginia Kaarthouer and issue date tomorrow.

Answer

```
int number = 123456;
String name = "Virginia Kaarthouer";
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.DAY, 1);
Date issueDate = calendar.getTime();
CreditCard card = new CreditCard(number, name,
issueDate);
```

Question

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

Constructors

100	main invocation	
	123456	number
	200	name
	300	calendar
	400	expiryDate
	500	card
200	String object	
	"Virginia Kaarthouer"	value
300	Calendar object	
	1415637372347	time
400	Date object	
	1415637372347	time
500	CreditCard object	
	600	issueDate
	700	expiryDate
600	Date object	
	1415637372347	time
700	Date object	
	1415637359054	time

Constructors

Question

Why can't the `issueDate` attribute of the `CreditCard` object not refer to the `Date` object on address 400?

Constructors

Question

Why can't the issueDate attribute of the CreditCard object not refer to the Date object on address 400?

Answer

If the the issueDate attribute of the CreditCard object were to refer to the Date object on address 400, then both the main invocation and the CreditCard object would have access to that Date object. But the CreditCard object "owns" that Date object, because CreditCard and Date form a composition. Hence, CreditCard should have exclusive access to that Date object.

Mutators

Question

Should we modify the mutator for `expiryDate` generated by eclipse?

Mutators

Question

Should we modify the mutator for `expiryDate` generated by eclipse?

Answer

Yes.

Mutators

Question

Should we modify the mutator for `expiryDate` generated by eclipse?

Answer

Yes.

Problem

Modify the constructor.