1

Summary of the last lecture

We have seen what is the UML notation, and applied it for class diagrams

It is now time to produce Java Code !

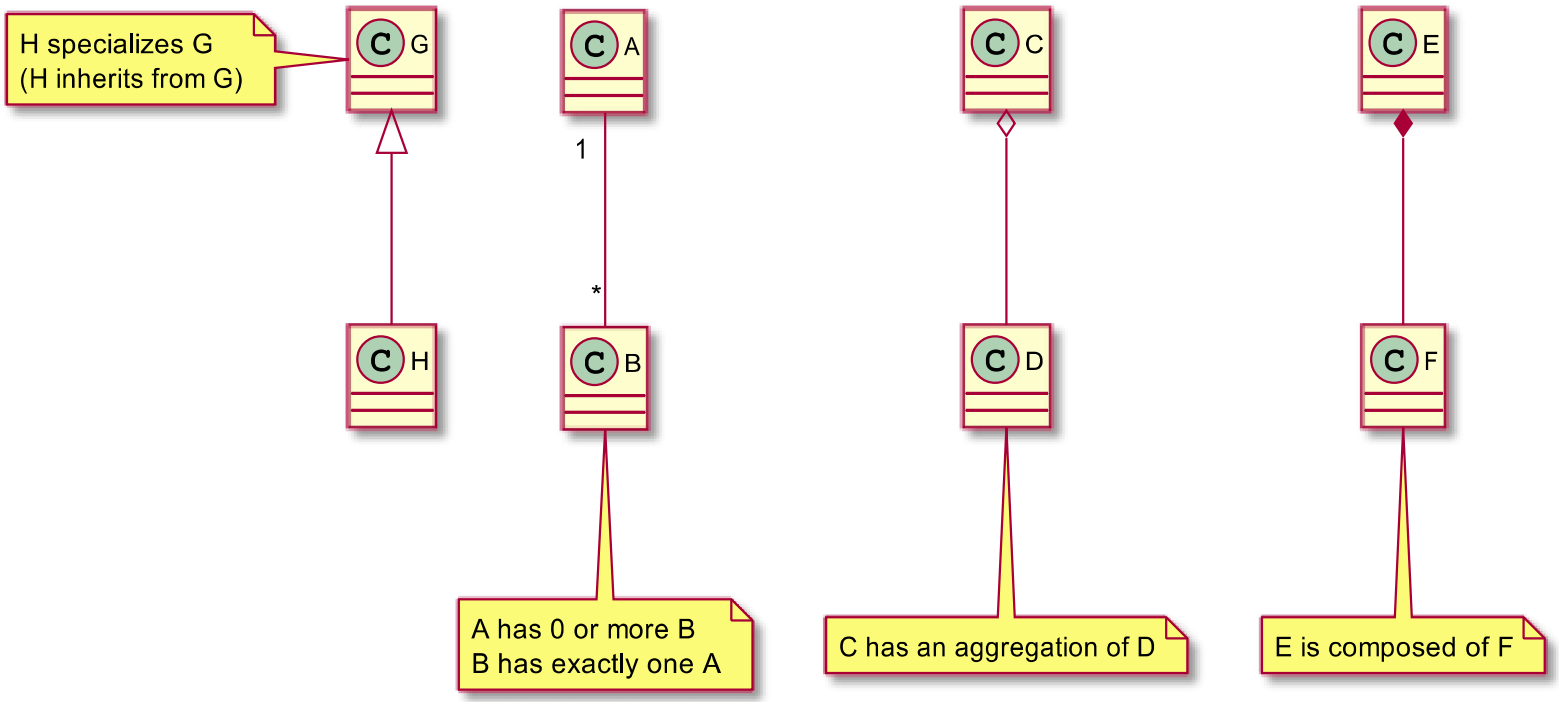
2

Summary of the last lecture

Who could define the 4 main concepts related to class relationship we've seen the last time?

3

Summary of the last lecture (2)



4

Our First Java Program : discovering Eclipse and Java

For this lecture we will need the Eclipse IDE

- Eclipse is the most used development environment for Java
- Eclipse can hold a lot of plugins, to match your needs

5

Using Eclipse

Eclipse global Overview :

- Eclipse is designed like a **Workbench**
- **Views :** Several views can be added to give other information on what you are editing
- **Perspective :** the workbench can scale to your favourite views organization
- **Workspaces** Several workspaces

6

Using Eclipse : Perspectives

As previously said, the views can be organized to increase your productivity

A certain arrangement of those views is called a **Perspective**

7

Using Eclipse : Projects

Eclipse has several project creation assistants

For now we will use the **Java Project** assistant

Let's begin a new Java Project !

8

Using Eclipse : Create a Java project

- Open Eclipse
- File > New > Java Project
- Call it "JavaExercises"

Notice that you can have other project creation assistant, depending on what **perspective** you are

9

Java : From Object Concepts to Java

We will create our first Java Class

10

Code analysis

- Comments

```
// This is a line comment

/* This is a block comment
it can contain several lines*/

/** This is a piece of javadoc, we will see this in detail after*/
```

- Statements

```
... ; //In Java, each statement must be finished by a semicolon
```

- The package declaration

```
package fr.tbr.exercises;
// It provides a unique namespace to the class, and helps to locate the class
```

11

Code analysis (2)

- Type concept: to tell that a field or a variable is of a certain type

```
Identity identity; // you should place the type name before the field
```

- **Constructors**, are special methods called to create a new instance of a class

```
//Constructor declaration
Identity(){
}

void test(){
    Identity identity = new Identity(); //constructor call thanks to the "new" operator
}
```

12

Exercise

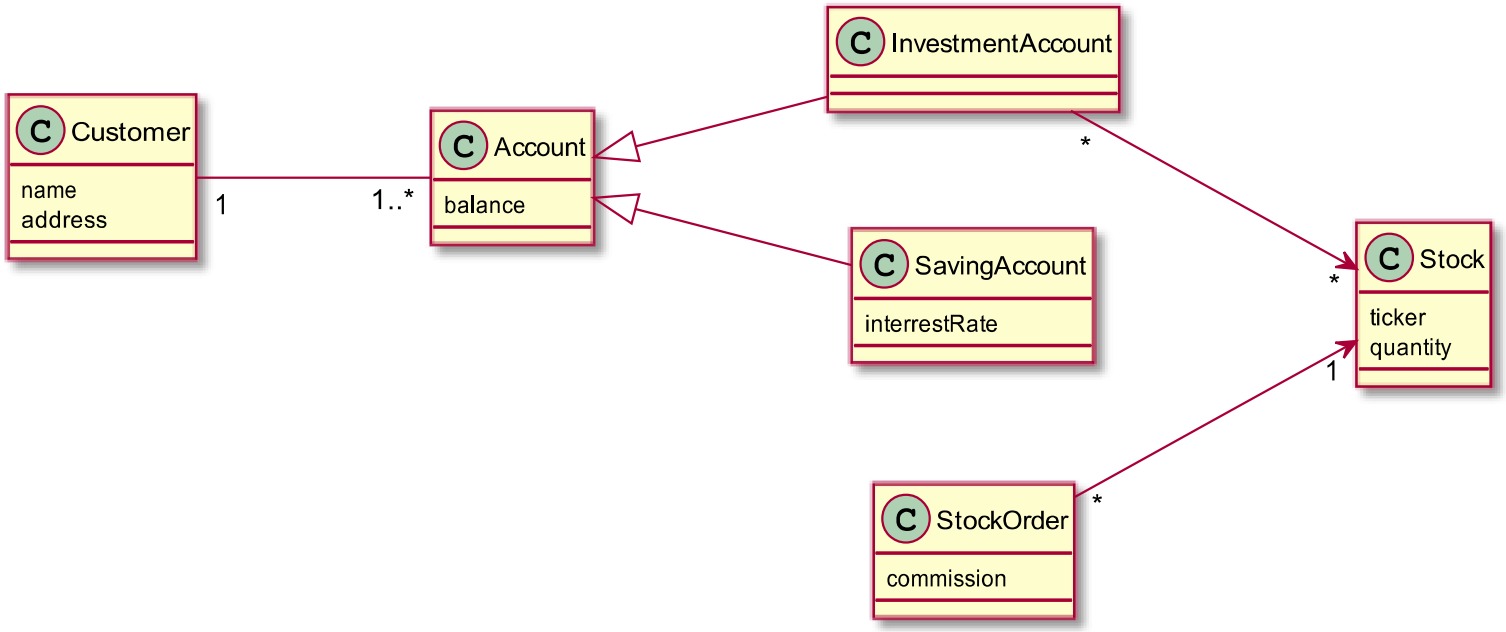
Remember the bank system? **Create each class in eclipse**

- Each "text" typed field should be represented as a `String`
- Each "numeric" typed field should be represented as a `double`
- Do not represent links between classes, the goal is only to create each classes
- Code help: `String` and `double` instantiations

```
String message = "Hello World";
double amount = 10.2;
```

13

Exercise (2) : Class diagram



14

Different kinds of Type

In Java, you can meet two "kinds of type"

- **Primitive types**, they are the builtin types of the language. They are not Objects
- **Object types**, they all inherit from the `Object` class. Your own objects belong to that category

Different kinds of Type: Primitive types

Type	Description	Range
`byte`	a signed byte	-128 to 127
`short`	This is a two-bytes signed integer, it defaults to 0 if not initialized	-32,768 to 32,767

16

Different kinds of Type: Primitive types (2)

Type	Description	Range
`long`	The long is a signed integer with a wider range. The declaration of a `long` is a bit different, you should do it as described below	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

```
long primitiveLong = 1222222333335555l;
```

Different kinds of Type: Primitive types (3)

Type	Description	Range
`float`	The `float` is a signed floating point number	1.4012984632481707e-45 to 3.4028234663852860e+38

18

Different kinds of Type: Primitive types (4)

Type	Description	Range
`boolean`	the boolean in Java can have two values `true` or `false`, which are reserved keywords of the language	`true` or `false`
`char`	2 bytes, unsigned, Unicode, `char` are used to represent characters, but are not directly compatible with integers or Strings	0 to 65,535

19

Numerical operators

There are several numerical operators :

- addition

```
int i = 0;
i = i + 1;
```

- subtraction

```
int i = 10;
i = i - 1;
```

- multiplication

```
int i = 10 * 2;
```

- division

```
int i = 10 / 2;
```

Numerical operators (2)

There are several numerical operators :

- modulo

```
int i = 25 % 2; // i equals 1
```

- increment

```
int i = 0;
i++; //i equals 1
```

- decrement

```
int i = 10;
i--; //i equals 9
```

Exercise

- Write a method `computeInterest()` in the class `SavingAccount`, that calculates the interest on one year, depending on the current amount at the computation time
- Write a method `withDraw()` on the same class, that takes one parameter