Programmation Orientée Objet – TP2

S.l. Alexandru Mitrea
S.l. Iulia-Cristina Stănică
iulia.stanica@gmail.com

Objectifs pour aujourd'hui

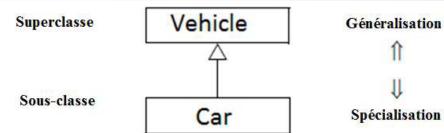
- Java recapitulation (suite)
- Héritage
- Polymorphisme
- Surcharge (overloading) vs. redéfinition / sous-typage (overriding)
- Lecture des fichiers

Héritage - Rappel

- Utilisé pour la relation généralisation / spécialisation (relation "est-un")
- Une sous-classe hérite / étend les **champs** et les **méthodes** (qui ne sont pas privés)

d'une superclasse:

- Ajoute de nouveaux champs / méthodes;
- Modifie (remplace) des champs / méthodes;
- Mot réservé en Java pour désigner l'héritage: extends
- En Java, une classe peut hériter une seule superclasse



Constructeurs

- Le constructeur de la sous-classe appelle toujours le constructeur de la superclasse, explicitement ou implicitement
- Pour éviter les erreurs de compilation, nous avons plusieurs possibilités:
- Dans le constructeur de la sous-classe, nous devons appeler explicitement l'un des constructeurs de la super-classe (en utilisant le mot-clé super)
- Insérez le constructeur sans paramètres dans la super-classe (car il est appelé implicitement par le constructeur de la sous-classe)

Polymorphisme

- = caractéristique d'une entité d'avoir des significations différentes, selon le contexte.
- Exemples de **polymorphisme statique** overloading (au moment de la compilation, dans la même classe):
 - constructeurs surchargés
 - des méthodes surchargées
- Exemples de **polymorphisme dynamique** overriding (au moment de l'exécution, dans des classes différentes):
 - méthodes redéfinies dans la sous-classe
 - la méthode toString() est définie dans la classe Object et redéfini dans nos classes.

+ MedicalStaff + Hospital Mettre en œuvre -noDoctors : int -name : String les classes -doctors : List -address : String +hospital: Hospital suivantes. +Hospital(name : String,address : String) +MedicalStaff(noDoctors : int,hospital : Hospital) +getName() : String +getDoctors(): List +getAddress(): String On suppose qu'un |+getDoctors(): List |+hireDoctor(d: Doctor): void médecin de famille +display() : void a un bonus de 120% et un chirurgien a un + Doctor bonus de 150%. -id : int -idCurrentDoctor : int #office : String #name : String #wage:double #bonus:double +Doctor(office : String,name : String,wage : double) +getId():int +setBonus(b : double) : void +treatPatient(): void +calculateWage(): void +toString(): String + FamilyDoctor + Surgeon +FamilyDoctor(office : String,name : String,wage : double) +Surgeon(office : String,name : String,wage : double) +treatPatient(): void +treatPatient(): void +calculateWage() : void +calculateWage(): void

Ex 1

Ex 1

• Supposons qu'on veut lire les informations d'un fichier externe (hospital.txt).

• Rappel lecture fichier:

```
FileReader fr = new FileReader("input.txt");

BufferedReader br = new BufferedReader(fr);

String strLine;

while((strLine=br.readLine())!=null){...}

br.close();
```

!!!Autre méthode utile: strlLine.split("separateur")

• Rappel ecriture fichier:

```
FileWriter fw = new
FileWriter("output.txt");

BufferedWriter out = new
BufferedWriter(fw);

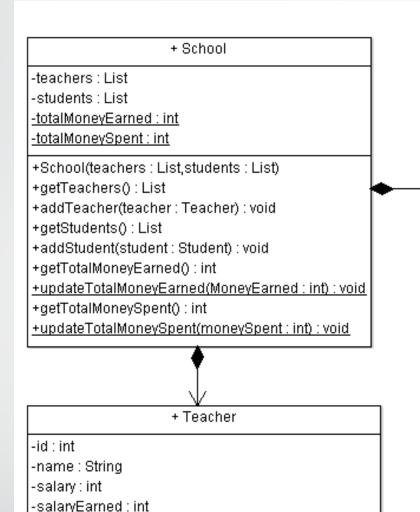
out.write("test");

out.newLine();

out.close();
```

Ex 2

- Mettre en œuvre les classes suivantes.
- Qu'est qu'on doit changer si on veut ajouter une autre classe, Personne?



+Teacher(id : int,name : String,salary : int)

+getId():int

+getName() : String +getSalary() : int

+toString(): String

+setSalary(salary:int):void +receiveSalary(salary:int):void

+ Student -id : int -name : String -grade : int -feesPaid : int -feesTotal:int +Student(id : int,name : String,grade : int) +setGrade(grade : int) : void +payFees(fees : int) : void +getId():int +getName(): String +getGrade(): int +getFeesPaid():int +getFeesTotal():int +getRemainingFees():int +toString(): String