THE UNIVERSITY OF SYDNEY

INFO1113 Session 5 Tutorial

Inheritance

Inheritance

Inheritance is a foundational feature in object oriented programming. Within java it allows classes to subtype other classes and **inherit** the properties of its parent type. This allows for greater code reuse as the subtype is able to inherit methods and attributes from its parent type that can be used within its own class.

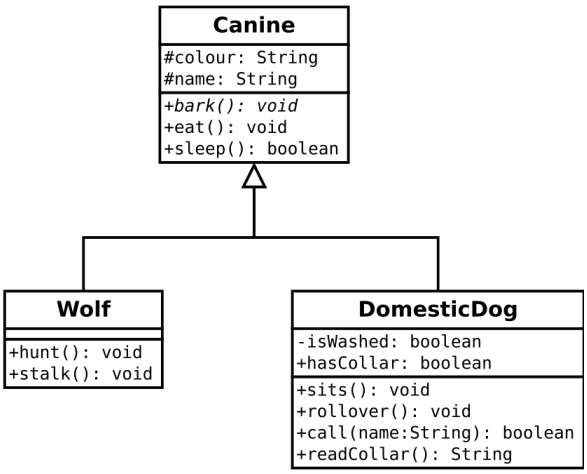

Figure 1: UML Class Diagram illustrating an inheritance heirarchy

As you may observe in the following class, that both `DomesticDog` and `Wolf` inherit from `Canine`. Any protected # or public + property is inherited by the sub-classes. This allows `Wolf` and `DomesticDog` access to `color`, `name`, `bark()`, `eat()` and `sleep()` methods from the parent class.

We would not have to rewrite any of these attributes for every subtype and any change that may occur will only occur in one place rather than multiple. The classes are also able to make a clear distinction between what properties define a `DomesticDog` in contrast to a `Wolf`.

Question 1: Cards

Consider the following attributes of various cards you might find in your wallet:

Credit Card:

- owner' name
- bank name
- card number
- expiry date
- security chip

Student Card:

- student name
- student ID
- year of issue
- magnetic strip

Driver's License:

- driver's name
- address
- card number
- state of issue
- license number

Part 1. What are the similarities between the cards?

Part 2. How are the cards used differently? How is data accessed from each of the cards?

Part 3. Discuss with your peers the the class definition below. What difficulties are going to be faced when using this class to represent all kinds of card (credit card, student card, license, etc.)?

Would this class be useful for maintaining a collection of `Cards`, for example in a Wallet?

```
public class Card {
        private String cardType;
        private String[] data;

        public Card(String cardType, String[] cardInformation) {
                this.cardType = cardType;
                this.data = cardINformation;
        }

        public String getCardType() {
                return cardType;
        }

        public void setCardType(String type) {
                this.cardType = type;
        }

        public String[] getInformation {
                return data;
        }

        public void setInformation(String[] cardInformation) {
                this.data = cardInformation;
        }
}
```

Part 4. Define a more suitable `Card` class that contains only the common atributes from each of the card types.

Part 5. Define a subclass for each of the card types described. These subclasses should use the `super` constructor from their parent class in order to initialise the common variables, as well as initialising their own specific variables.

Question 2: Critique

One of your peers has given you a UML Class Diagram to criticise. Discuss with your class, tutor, friend or dog about the issues with the design.

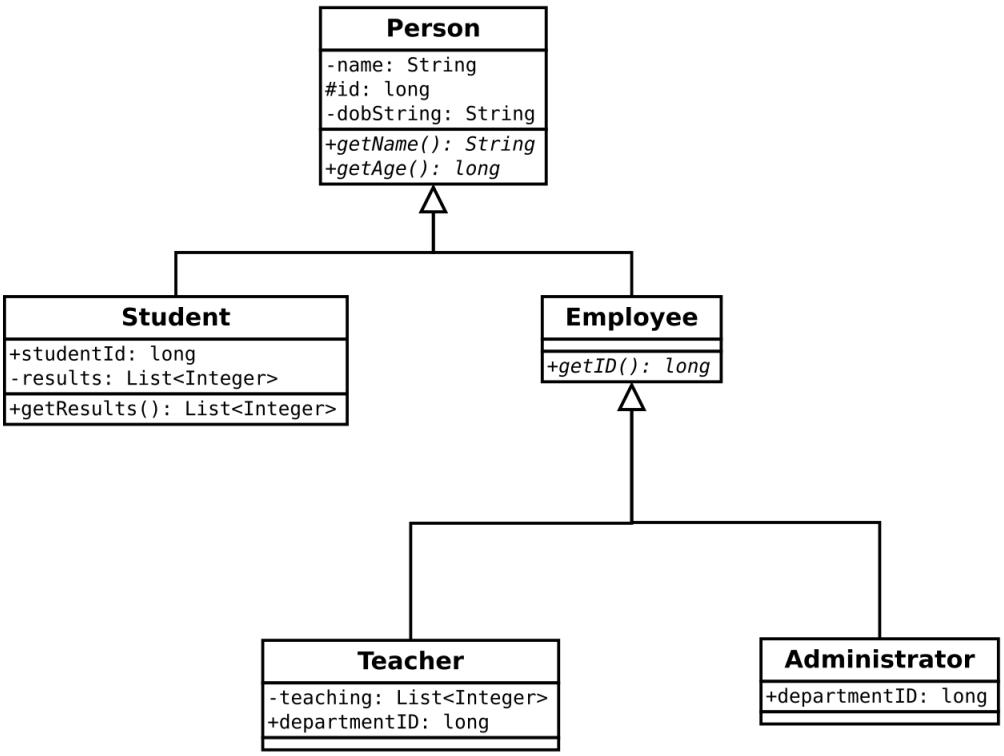

Figure 2: Your peer's work

- What attributes are not inherited?
- Outline the redundant attributes from the class diagram
- What properties of the classes would you fix?

Question 3: Bank Account

A bank account has an account number, BSB, and interest rate. The account can be opened with an initial balance. It can perform several transactions: earn interest, withdraw, deposit or transfer to another account, and can generate an invoice for a given time period.

A transaction has a date, a text description, and an amount (positive for a credit, negative for a debit).

An invoice has a date period, a statistical summary of bank transactions and attributes of the bank account.

For example, the following bank account:

```
Account number: 123456789
BSB: 123456
Initial Balance: \$0.35
Interest Rate: 1.5\% p.a.
```

Your program could generate the following invoice:

```
Account number : 123456789
BSB : 123456
Interest rate : 1.5%

Statement for 01/05/2012 to 31/05/2012

Date:       Details:                Debit:      Credit:    Balance:
01/05/2012  Opening balance                     $0.35      $0.35
03/05/2012  Salary Crazy clown airl.            $451.00    $451.35
06/05/2012  HD Max virtual cinema   $30.00                 $421.35
06/05/2012  HD Max virtual cinema c. $8.00                 $413.35
06/05/2012  Hip froyo               $16.10                 $397.25
16/05/2012  Salary Crazy clown airl.          $529.00     $926.25
32/05/2012  Account interest                  $1.16       $927.41
```

Part 1. Describe (not code!) the above problem using classes. You should describe at least three classes: `Account` , `Transaction` and `Invoice`.

Part 2. Describe as many class invariants for these classes as you can.

Part 3. How is information passed between a bank account and its invoice?

Part 4. Write a program to create a bank account, passing it all necessary initial values.

Part 5. Add methods to describe a transaction being performed on the bank account.

Part 6. Add a method to generate a invoice for a given time period, print the statement with the format given in the picture.

Part 7. implement a subclass of `Account` called `SaverAccount` . For this type of account, you can earn a bonus interest rate of 2.3% p.a. above the base interest rate of 1.5% when you grow your balance by at least $500 by the end of the calendar month.

Question 4: SimpleDate

Create your own date class `SimpleDate` to store date information in your program for question 3. Complete the following methods for the correct functionality.

```
public class SimpleDate {
        private int d, m, y;

        public SimpleDate(int d, int m, int y) {
                this.d = d;
                this.m = m;
                this.y = y;
        }
        // return true when this date < other date
        public boolean isLessThan ( SimpleDate other ) {
        // complete this code
        }

        // Repeat for the following :
        public boolean isEqualTo ( SimpleDate other ) { ... }
        public boolean isLessThanOrEqualTo( SimpleDate other ) { ... }
        public boolean isGreaterThan ( SimpleDate other ) { ... }
        public boolean isGreaterThanOrEqual( SimpleDate other ) { ... }
}
```

To use the `SimpleDate` class, for instance, one method in your program will have the following header.

```
Invoice generateInvoice (SimpleDate startDate, SimpleDate endDate)
```

Question 5: Assessed Task: Quiz 1

Remember you are required to complete the quiz next week. Go to Canvas page for this unit and click on Quizzes to find out the quiz and the due date. This is a marked assessment.