SOFT1002
Week 2: Inheritance

School of Information Technologies

Today’s Lecture

◆ Inheritance
◆ Using Abstract Class
◆ Static and Dynamic Type
◆ Using Interface
◆ Abstract Class versus Interface
◆ Read Big Java Ch.11

Announcement

◆ In this week,
  ● You should have formed the group for Task A.
  ● Your group should have a structural prototype and be able to demonstrate to your tutor this week.
  ● Your group should come up with a group ‘coding’ standard.
  ● The group should prepare the test cases for your system NOW (yes, even before you start your first line of code!)

Announcement

◆ Anyone who
  ● Got a Distinction or above in Soft1001/1901 can enrol to Soft1902
  ● Failed Soft1001/1901 should drop this subject and repeat Soft1001 again

Why use Inheritance?

◆ Save time to type in common features 😊
◆ Code reuse is more efficient
◆ Help minimise error when changes have to be made
◆ Enforce a class design

Let’s look at inheritance using the example of:
DiceGame & CoinsGame

◆ super
◆ Inheriting fields
◆ Inheriting methods
◆ Overriding methods
◆ Adding new methods
**Abstract Class**
- Details of a method is deferred to the subclasses using the keyword `abstract`.
- There is no body in an abstract method when it is declared, e.g.,
  ```java
  public abstract void draw();
  ```
- No object can be created from an abstract class.
- Abstract classes enforce the class design.

**Let's look at the Abstract class using the example: Animal**

**Static and Dynamic Type**
- Static Type:
  - The type the variable is being declared.
  - Known at compile time.
  - The word 'static' here has *nothing* to do with the static keyword in Java.
- Dynamic Type:
  - The type of the object that a variable is actually referencing.
  - Known only at run time.
- Example,
  ```java
  Account acct = new SavingsAccount();
  ```

**Let's look at the Static and Dynamic Type using the example: Pet**

**Example of Static and Dynamic Type**
```java
Account myAcct;
myAcct = new SavingsAccount(); (A)
myAcct = new ChequeAccount(); (B)
```
- `myAcct` at (A)
  - Static Type: Account
  - Dynamic Type: SavingsAccount
- `myAcct` at (B)
  - Static Type: will Account
  - Dynamic Type: ChequeAccount

**Polymorphism**
- Also called Dynamic Binding
  - Binding is done during run time, rather than at compile time.
  - A variable has a static type, but when you call a method, it is the dynamic type (the actual type of the object referenced by the variable) is used to determine which actual method should be called.
What is an Interface?

- Java does not allow multiple inheritance, i.e. to extend from more than one class.
- Interface only has method signatures.
- A class can implement an interface.
- A class can extend from one class and implement multiple interfaces.

Abstract Class versus Interface

- They are both used to enforce a design.
- Abstract class can have concrete methods but interface does not.

What have we done?

- Inheritance, c.f. DiceGames & CoinGames
- Using Abstract Class, c.f. Animal
- Static and Dynamic Type, c.f. animals in Pets class
- Interface
- Abstract Class versus Interface

- Prepare for next week: Ch.14 and 15.1-2