- Java is a strongly typed language which means every variable has a type.

// if manager extends employee:

Employee e = new Manager(); // this is fine

Manager m = new Employee(); // this is wrong, will not compile

Instance of and class casting

- instanceof is an operator

Employee e = new Manager();

if (e instanceof Manager) // if the opject e it is referring to is actually of manager type, it will return true

Manager m = (Manager)e // class cast

Polymorphism

- the object e before it is cast to Manager can only call methods from the Employee class

- objects only can call methods from the class of the object’s type.

Example:

Animal a = new Cat(); // assume Cat extends Animal

a.doX();

- Problem solving strategy: check left hand side then check right hand side

- if animal class contains a method called doX() and cat class contains doX(), then this code will compile successfully and the doX() in the cat class will be invoked.

- if animal class contains a method called doX() and cat class does not contain doX(), then this code will compile successfully and the doX() in the animal class will be invoked.

- if the animal class does not contain a doX() method, there will be a compile error.

- all the overriding rules apply here when comparing the 2 methods

- when you are checking the left hand side, check all the way up the inheritance tree

Example: if animal has a parent class mammal check the methods in mammal also.

- when you are checking right hand side, the method that compiles is the most sub class.

Example: if cat has a subclass lion, and both have dox() methods, cat.doX() will invoke the doX() in lion.

Abstract Methods

- an abstract method cannot be called.

Example:

public abstract void displayInfo();

- a concrete class CANNOT have abstract methods.

- in uml diagram, abstract class names are italicised

- you cant create an instance of an abstract class, but you can declare an object reference variable of that type

- it is possible to extend a concrete class or an abstract class to create an abstract class

Interface

- contains any no. of abstract methods

- doesnt contain any concrete methods

- all attributes are public static and final

- like an abstract class you cannot instantiate an interface, but you can declare a varible of that type

- when you implement an interface, all the abstract methods get inherited by the child class.

- you can implement multiple classes

- any variable that you define within an interface is not an instance variable. it is static and final.

Equals overriding

- first check if the object type is the same

- then check if the attributes of the object are the same

- return true if everything checks out

- The comparator class is an interface with the <> sign