JA111

Day-8

Super class reference variable and child class object

An object of subclass has everything that an object of super class has so object of subclass can be considered as object of super class.

A reference variable of super class can point to the object of child class but using that reference variable we can access only those members that are defined in super class because reference variable of super class do not have any knowledge about the members defined in subclass.

Run time polymorphism/Dynamic polymorphism/Late Binding

Using method overriding, we are redefining the behavior of general class into the specialized class i.e. it is used to define the general behavior of a super class in the sub class according the specialized need.

A super class ref. variable can point to child class object and this ref variable can access methods defined in super class only. The overridden method is defined in super class and sub class both so the overridden method can be called using ref variable of super class such that

a. If reference variable is of superclass and object is of super class then version defined in super class will be called.
b. If reference variable is of superclass and object is of sub class then version defined in sub class will be called.

The calling decision of overridden method is taken at run time that's why it is a form of run time polymorphism. The concept of calling the overridden method at runtime is called run time polymorphism.

Miscellaneous about Method overriding

When overriding a method in subclass make sure to use same or less restrictive access modifier, never use more restrictive access modifier because you cannot downgrade the access modifier but you can upgrade.

private (Most restrictive) < default (no modifier) < protected < public (Least Restrictive)

Overridden method can return same or subtype but it cannot return super type. This rule is called covariant return type.

instanceof operator

It is used to check if the object belong to the specified class or not

Syntax: obj-name instanceof class-name

Up-casting & Down-casting for reference types

The up-casting is automatic i.e. if object/reference of sub class is assigned to reference of super class then this conversion is automatic.

The down-casting is not automatic i.e. if object/reference of super class is assigned to reference of sub class then this conversion has to done using typecasting otherwise compiler will produce error. Yet after manual typecasting run time error (that is ClassCastException) will be generated if sub class reference variable points to object of super class.

Overriding Methods of Object class

In java we do not have operator overloading so meaning of == cannot be redefined; if you wanted to check if two different references point to same object then use == but method overriding is available so meaning of equals() can be redefined; if you wanted to check if two objects has same content (may be for all fields or for one field) then override equals method and use it.

The default implementation of equals method uses == so if two references point to same memory location then return true and hashcode for both references will be same. So when override equals method make sure to override hashcode also such that for two objects if equals method return true then their hashcode must be same.

If you are overriding the hashCode method then it is not required to override equals() method because for two different objects the generated hashcode may be same.

The above both points are convention not rule so if you do not follow then no error will be there but for long run/correct implementation follow both because you see its impact in collection framework.