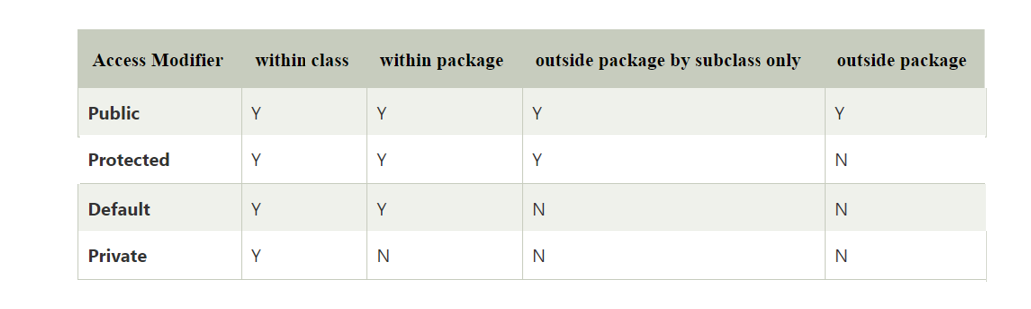
Access modifiers:

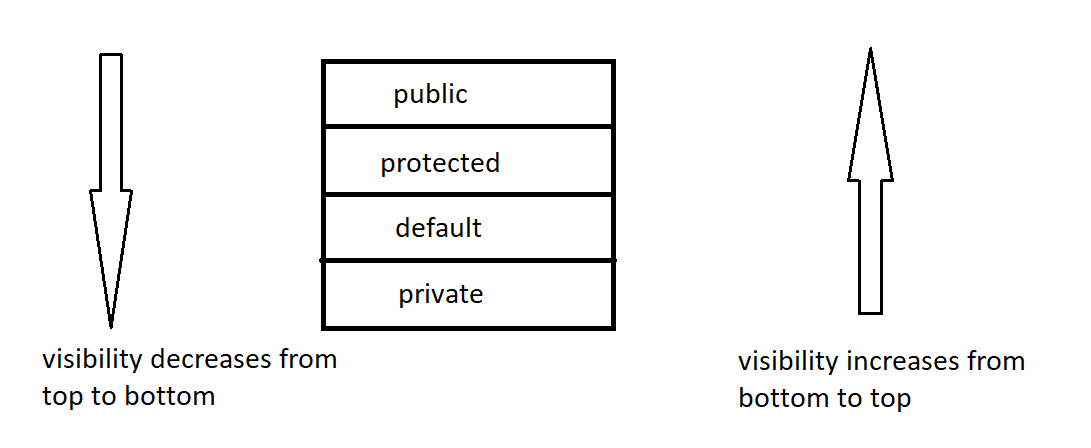
Access modifiers can be applied on class, variable (can be used only on static variables, instance variables. Local variables cannot use access modifiers, since their scope only within their methods (or) block (or) loops (or) conditional statements) constructor and method.

They cannot be applied on static block and java block.



Note: we cannot declare a top - level class as private (or) protected.

// work on this after packages.



Note:

1. Methods which are inherited from parent, and child is using that methods without any change they are called as inherited methods.
2. Methods which are inherited from parent, and child is making some changes to that method they are called as overriding methods.
3. Methods which are present in the child class, but not present in parent class is called specialized methods.

Overloading: when two or more methods in same class having same name but different parameters is called overloading.

Overriding: when method signature (name and parameters) present in parent class and child class are same then it is called Overriding.

Note: overloading can be done in same class, it is achieved in inheritance also.

Overriding cannot be done in same class, it throws compile time error. It is achieved only during inheritance.

Eg: Inheritance\_Method\_Types

// go through the code

// practice uml for better understanding.

Rules to override method:

1. We cannot decrease the visibility of overridden method. But we can increase the visibility of overridden method.

Note: overriding method is present in parent class.

Overridden method is present in child class.

1. Return type of overriding method and overridden method must be same.

Note: For methods the return types can be class type also. if return type of method is class, the method should return object of that return type class.

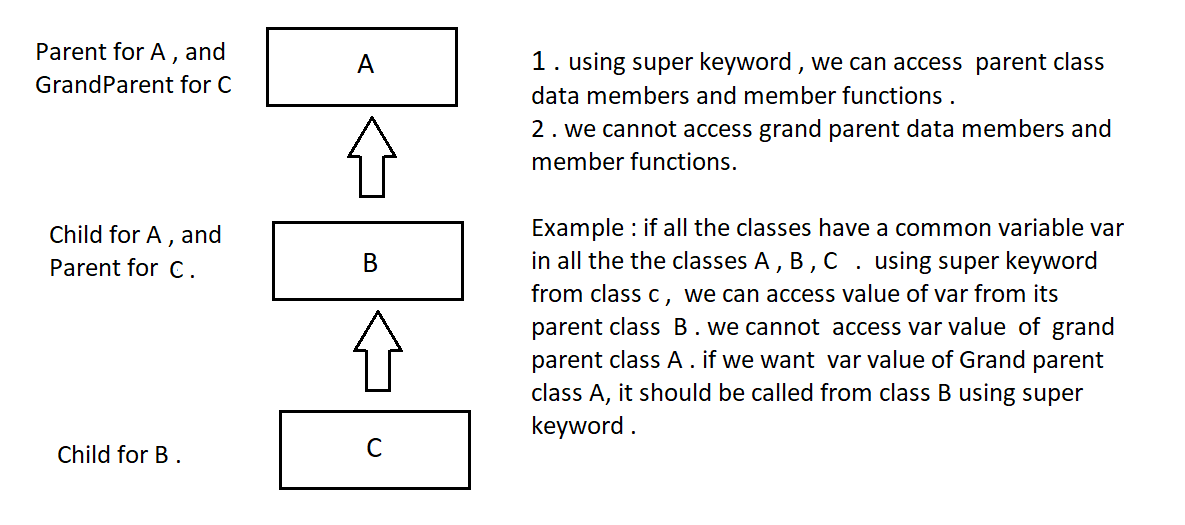
1. If there exist is-a relationship between returns types of overriding method and overridden method, then return types of overriding method and overridden method can be different. This is called co-variant return type.

Eg: Co\_Variant

1. Parameters of overridden method must be same as that of overriding method, else it will treat as specialized method (considered as overloading).

Eg: Override\_Rule\_4

Note: super keyword is used to call parent class instance variable, if name of both variables is same.



Eg: Super\_Keyword\_Inheritance

Note: this keyword is used to access current class data members and member functions. while super keyword is used to access parent class data members and member functions.

final keyword

1. It can be applied to class, variable, method.
2. If a class is made final it cannot be inherited.
3. If a method is made final it can be inherited, but we cannot override it.
4. Final variable acts as constant we cannot change that value it participates in inheritance, but value can’t be changed.

Polymorphism is divided into two types.

1. Compile time polymorphism achieved by method overloading.
2. Runtime time polymorphism (or) true polymorphism achieved by method overriding.

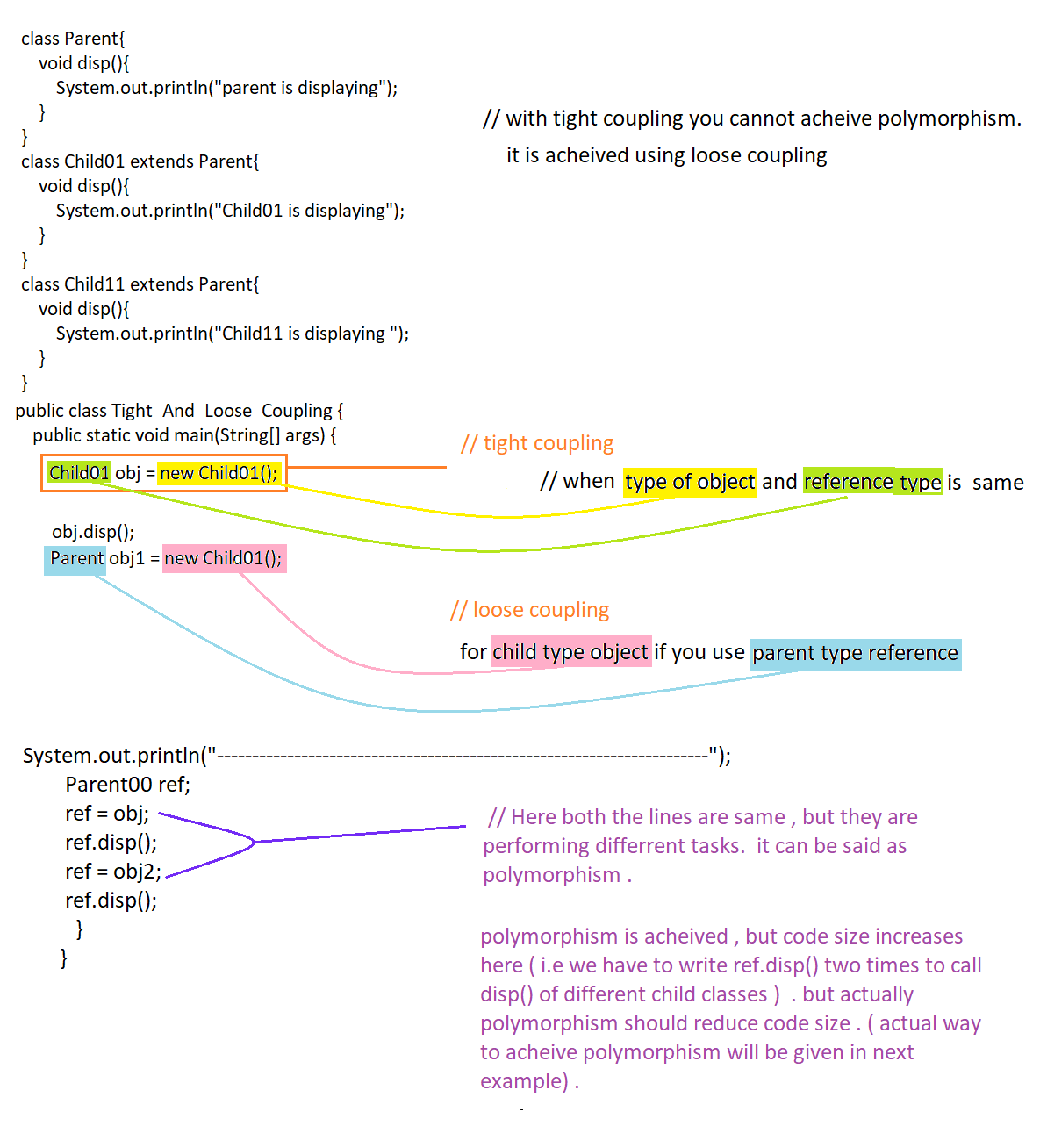
Note: By inspecting the method signatures, compiler determines the method to invoke at compile time. As a result, this is known as compile-time polymorphism, static polymorphism, or early binding.

Call is made to the method by the jvm based on the runtime object during runtime so it is called runtime polymorphism.

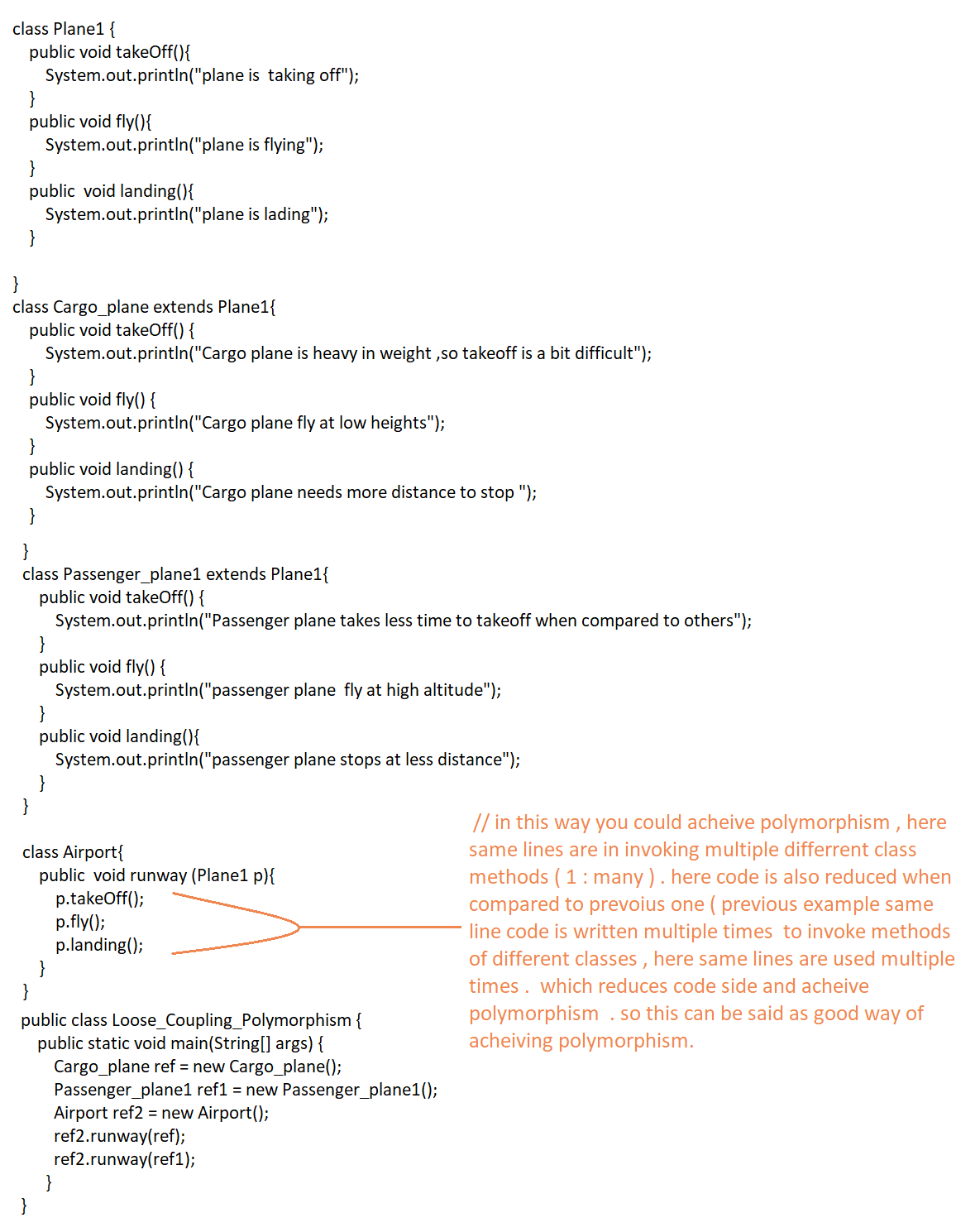
Advantages of runtime polymorphism:

1. Flexibility
2. Code size reduces

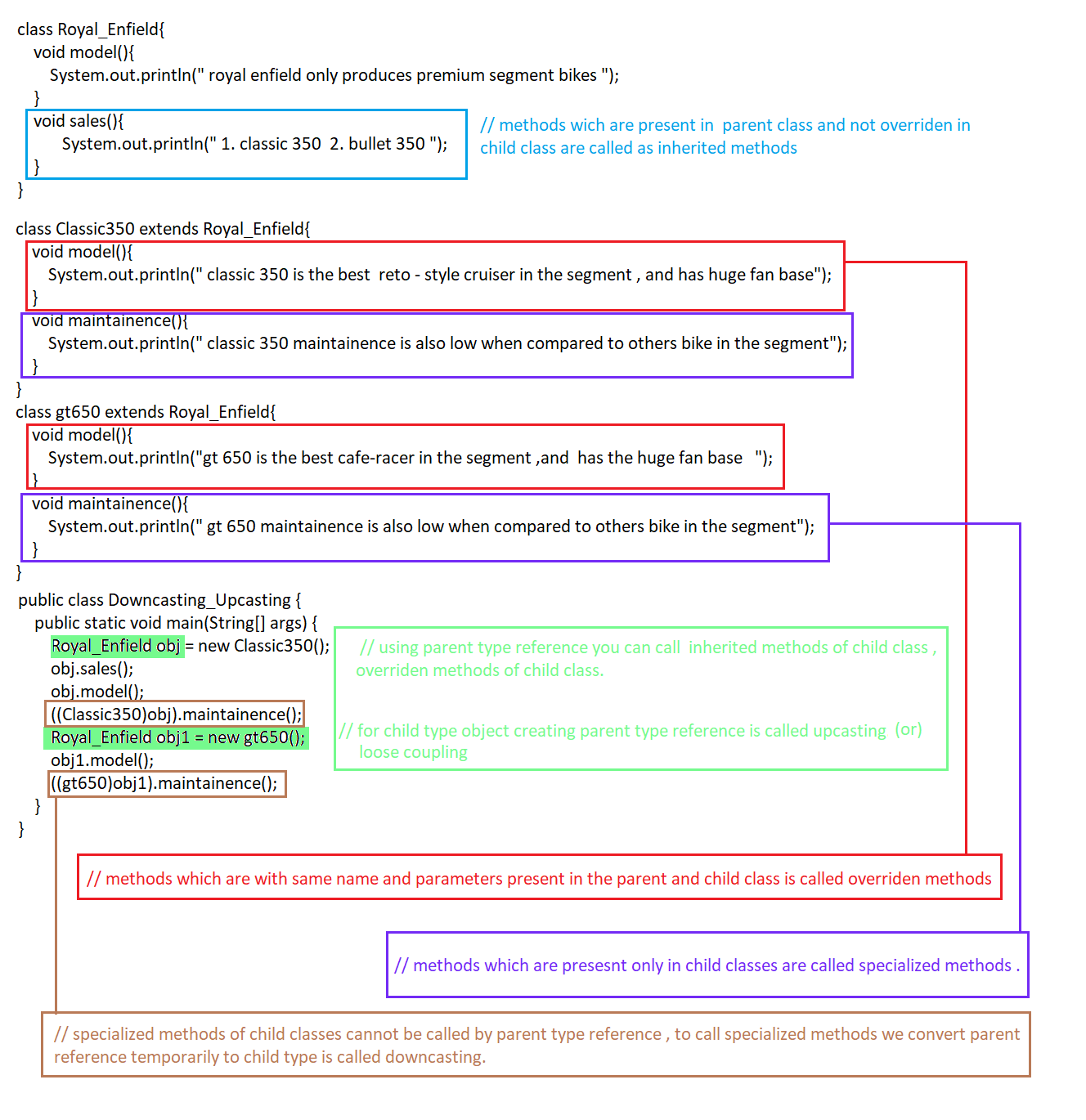
Eg: Tight\_And\_Loose\_Coupling



Eg:Loose\_Coupling\_Polymorphism



Eg: Downcasting\_Upcasting.



Method Hiding:

Using parent type reference with child class object, we can access inherited methods and overridden methods. but we cannot access specialized methods

Static methods do participate in inheritance

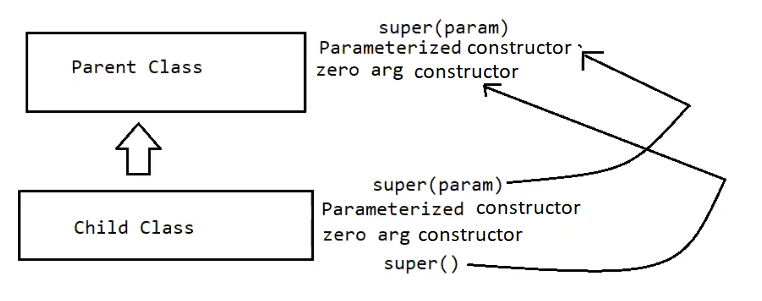
Eg: Static\_Methods\_Do\_Participate\_In\_Inheritaance

Static methods do participate in the inheritance, and gets inherited in the child class. But if you try to override the parent class method in the child class method then, it will we treated as specialized method. this concept is called method hiding

This mechanism happens because the static method is resolved at the **compile time**. Static method bind during the compile time using the **type of reference not a type of object**.

Eg: Method\_Hiding

Eg: Inheritance\_Eg1

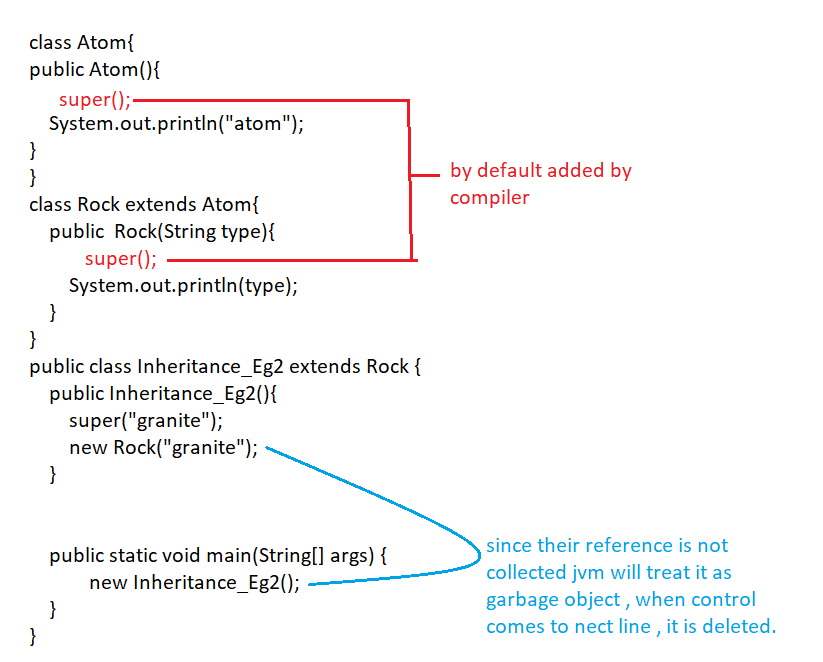


Super() will be included in the first line of the child class constructor by default by compiler. The super method calls the parent class constructor. but in the parent class if user writes the parameterized constructor, the compiler will not include default constructor.

But jvm searches for the zero parameterized constructor, since compiler included super() with no arguments by default. which is not present, so it leads to compile time error.

to solve this problem include zero parameterized constructor in parent (or) make a call to parameterized super() method manually in the 1st line of the child class constructor.

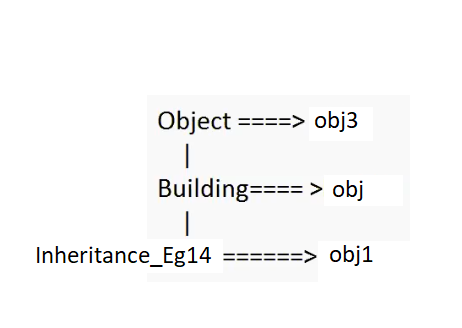
Eg: Inheritance\_Eg2



Eg: Inheritance\_Eg3

// go through the code

Eg: Inheritance\_Eg4



When you do the typecasting there should be a relationship between parent to child (or) child to parent.

Child object can be collected by parent type automatic it is, but if you want parent object to be collected by child object then you should type cast.

Eg: Inheritance\_Eg5

Here the Money class has a private property and getter method, so we can say that class as encapsulated class.

Anyone want the private property of Money class they should create object of money class and access the getter method

In class Yen we are extending Money and overriding getC() method no problem

In method getC() we are using super keyword to access the private property of the parent , which is not possible leads to compilation error . it is possible only when property of the parent is public.

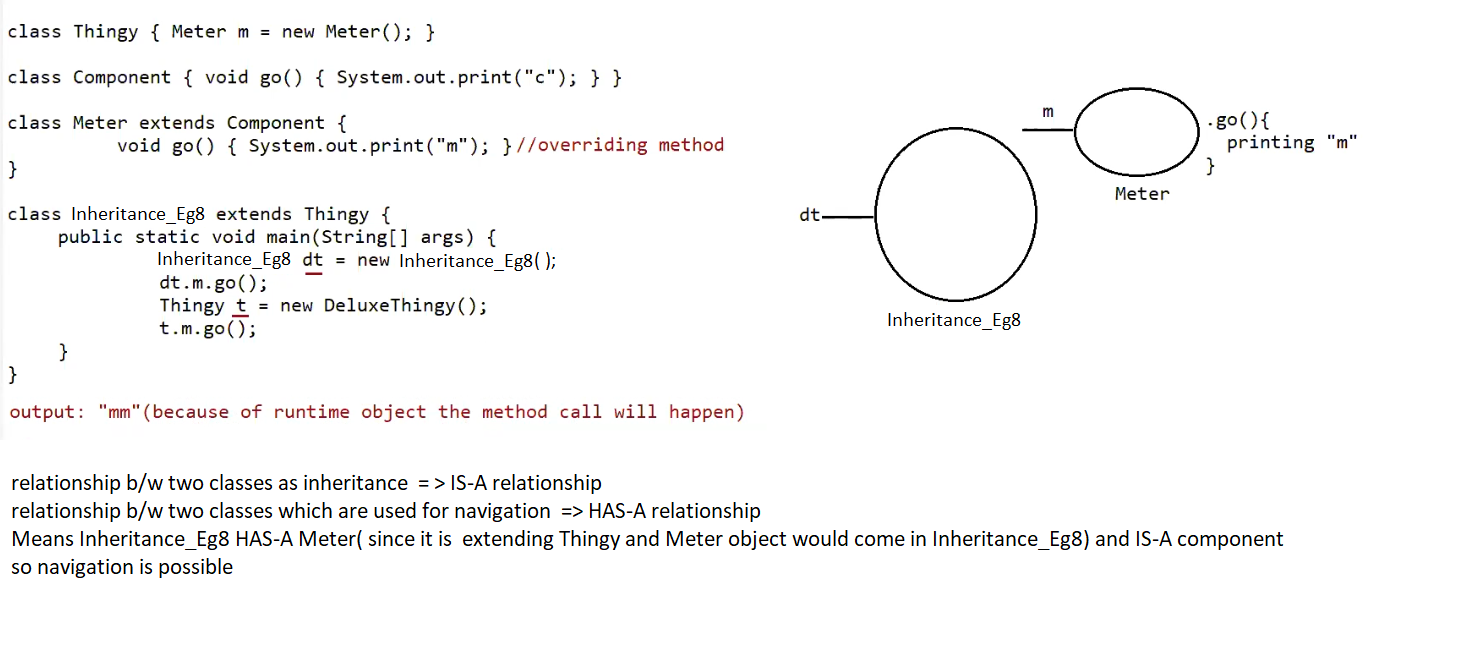
Eg: Inheritance\_Eg6

// go through the code

Eg: Inheritance\_Eg7

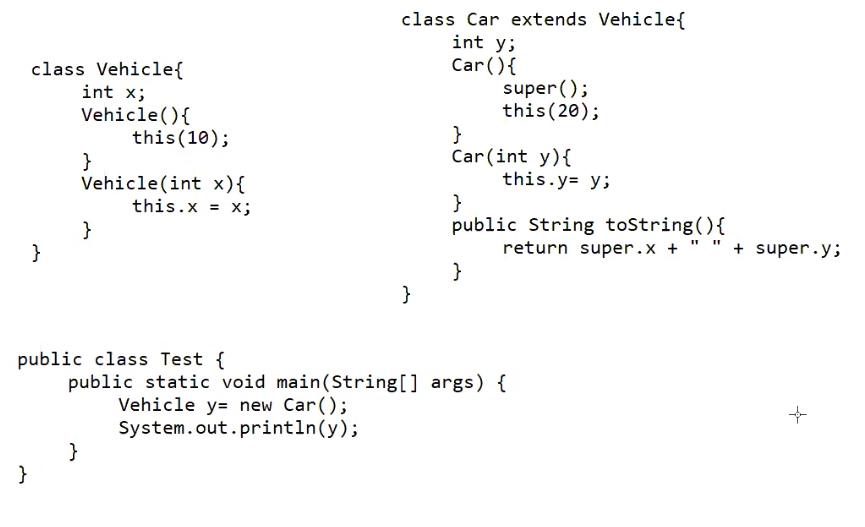
// go through the code

Eg: Inheritance\_Eg8

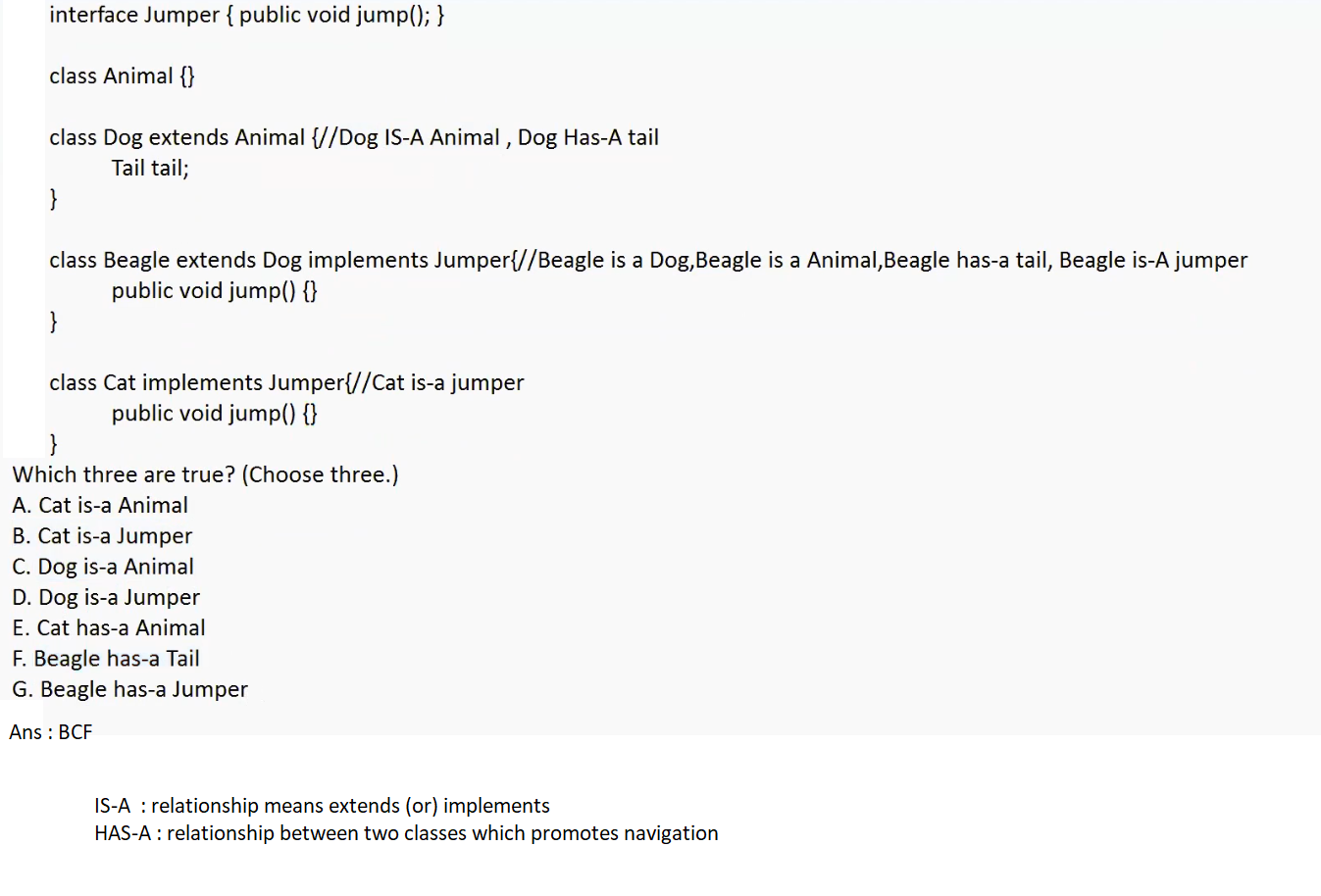


Execution starts from main(), object for Inheritance\_Eg8 will be created and its default constructor is called. The default constructor of Inheritance\_Eg8 has super() included by compiler . now the control goes to the parent class constructor i.e Thingy() . now memory for instance variables of Thingy is allocated. (memory for meter is allocated).

After that using reference dt and m we are calling go(). here go() of meter class will be executed because of runtime object the method call will happen to the overring method .



super() and this() should not present in the same constructor , it leads to compilation error.



Eg: Inheritance\_Eg9

// go through the code