

### Inheritance

#### Why inheritance?

1. DRY(Don't Repeat Yourself): No copy and paste, use has a (composition), is a (inheritance)
2. Extensible: easy to add/modify business logic and share the code)

#### Java inheritance

Single inheritance, one parent only , all instance variables and methods inherited  
one parent could have multiple child classes  
parent/super/generic <- child/sub/specifc

### Chaining constructor

#### Chaining constructor

child constructor call parent constructor  
reasons 1: private parent instance variable  
reasons 2:clean and neat:compare: loosen restrict, add using setter(how about set is not logically OK)

super() is always called explicitly()or implicitly in the first line of child constructor.  
-> if a class will be extended, it must has no argument constructor, or do not have any constructor.

super.xxx(dot operation on super) won't follow first line rule of constructor.

#### More on protected

Package private + subclass

parent/child + different packages, access protected state/behavior from parent	1) direct call 2)obj ref of <b>the child</b> itself 3) obj ref of parent or other sub class can NOT.
--	--

### Overriding

in child change behavior of parent

1. same signature as parent
2. return type, same or subtype, primitive exact the same(no promoting and wrap)
3. accessible: same or wider
4. exception: same or fewer/subtype/runtime

private/static methods are hidden, not overridden	polymorphism applies only instance method
---	---

super. ( dot notation) to access parent state or behavior

never hiding static member ( variable/-method) or instance variable, bad practice, confusing.

### Covariant returns

overriding method return a same or subtype of parent returned

exact the same for primitive return

### Three Faces of Final

<b>final variable</b>	Constant
enum constants	constant used in implicitly static final
<b>final method</b>	no overriding
<b>final class</b>	no extends, java.lang.String

Switch: literal,constant,enum, compiler time bind, variable or method return not, due to not known how many cases should be listed.

### Class/Object invocation order

1 static var=default	child->base
2 static {}, explicit value assign to static var	base, in statement order
repeat 1,2 in child hierarchy order	->child
4 instance var=default	child->base
5 {},explicit value assign to instance var	base, in statement order
6 constructor	base
repeat 4,5,6 in child hierarchy order	->child
static initialization 1-3 execute only once when first class is loaded.	

### Overloading

same name+diff-erent parameters	Signature=name+param,unique in a class
---------------------------------	--

Others(return type,modifiers, exceptions) not matter

overloading matching order:

exact match ->promoting->wrapper =>v-arargs (exact match,promoting,wrapper)

### Private methods redeclaration

not inherited

can redeclare a method with same signature

### hiding static method

with static parent method, could not override

hiding - no polymorphism

4 overriding rules + static modifier

Never hiding static methods in practice, confusing and bad habit



By Jianmin Feng (taotao)  
[cheatography.com/taotao/](https://cheatography.com/taotao/)

Not published yet.  
Last updated 5th May, 2019.  
Page 1 of 3.

Sponsored by [ApolloPad.com](https://apollopod.com)  
Everyone has a novel in them. Finish Yours!  
<https://apollopod.com>

### Inherit variables

never overriding, always hiding if same name

when hiding a variables, using super and this to access parent and child

static and non static follow the same rule for hiding

private variables inherited but could not access directly.

never hiding variables in practice, confusing and hard to read code

### Abstract classes

#### Why?

generalization, inheritance, overriding and polymorphism

simply code, beauty, no DRY

prevent improper instantiate of parent classes

#### Abstract class rules

>=0 abstract methods

can't initialized

public /package private only, must be extends, so private or final is not allowed, protected is not logic/meaningful

extends abstract class means overridden all abstracted methods or declared as abstract

first concrete class must have implemented all abstract method directly or indirectly

#### Abstract methods

in abstract class

can not be private, final, static (must be overridden)

no body, even {}

### Abstract classes (cont)

overriding rules(4) must be followed: same signature, broader or same visibility, narrower or same return type, narrower or same exception throws/or runtime exception

### Interface

public abstract interface ....{}

public static final init at the statement  
MIN\_DEPTH=3

interface extends multiple extends  
interface1,interfa- allowed here  
ce2,...

class impluments multiple implements  
interface1,interfa-  
ce2,...

can redeccclare a method with same signature

#### Rule for interface

can not instan- may have no methods  
tiated at all

public / default not private,final,  
only protected for interface

all methods must not private,final,  
be public protected for methods

abstract method in java8, default, static  
by default methods with a body  
allowed

**default interface** java 8  
**methods**

mainly for backward compatibility

public default double calc(){}

### Interface (cont)

only in interface can be redeclared as  
abstract or overridden  
with a different body

not static, not private,protected  
final,or abstract  
( overridden)

#### Multiple inheritance problem

default method in interface, if not overridden,  
will cause compiler error if default methods  
with same signature existed

for interfaces without default methods, there  
is no this issue

if default method is overridden, also no  
ambiguity problem.

**Static method** java 8 above

public or default with a body  
only

call with this avoid ambiguity  
interface name, cause be multiple inheri-  
not with object tance  
ref

static methods in interface could be  
declared as default in sub interface

designed to offer utility functions

protected method in interface does not  
makes sense as has nothing to be shared  
with the subclass. it is just an interface.  
multiple inheritance of type:

<https://docs.oracle.com/javase/tutorial/java/landl/multipleinheritance.htm>

<https://www.baeldung.com/java-static-default-methods>

<https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-java/>



By **Jianmin Feng** (taotao)  
[cheatography.com/taotao/](https://cheatography.com/taotao/)

Not published yet.

Last updated 5th May, 2019.

Page 2 of 3.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish  
Yours!

<https://apollopad.com>

### Polymorphism

heart of inheritance (overriding+polymorphism) separate of concern, flexible/extensible coding

properties of an object to take on many different forms, compiling time- ref by super class/interface ref, at run time multiple behavior, based on the object itself

multiple references (on the stack) (ref of type of super class, interface), static binding multiple object (on the heap) behaviors, dynamic binding

**Virtual methods** dynamic method dispatching

overridden methods non private, static, final

a method in which the specific implementation is not determined until run time; at compiling time, parent ref is used, at run time, implementation based on the child obj referenced

### Object casting

implicit up casting (child -> parent)

explicitly down casting (parent -> child)

error for non-parent/child object casting safe casting: obj1 instanceof obj2

### Polymorphic parameters

parameter is parent class or interface type

### Polymorphism (cont)

passing the child obj or obj implemented the interface auto up casting

a reference variable may only send messages that are available to its type. being available to an object != being declared inside an object

### when Parent/Child not belong to same package

#### in extending class

protected parent members could be accessed directly. if by references, only by the ref of extending class itself

ref variables of other child class or even the parent class COULD NOT access parent members inside extending class

#### constructor()

if constructor missing access modifier (package private), the child class could not be instantiated.

if class to be extended, constructor must be public or protected, if in different package.

### Pass by value vs by reference

Both are passing by copy

the original content (primitive value or object memory address) variables not affected

if passing a copy of obj address, changes to the object on the heap will be shared with all references.

reassign the reference in callee will not affect the ref in caller

variable on stack frame, obj on heap

