Cheatography

Java Class Design (OCA) Cheat Sheet

by Jianmin Feng (taotao) via cheatography.com/79308/cs/19394/

Inheritance

Why inheritance?
1. DRY(Don't Repeat Youself): 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 inheritated
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 **the child** 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 overriden	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

final variable	Constant
enum constants implicitly static final	constant used in switch
final method	no overriding
final class	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+different parameters	Signature=name+param,unique in a class
Others(return type,modifiers, exceptions) not matter	
overloading matching order:	
exact match ->promoting->wrapper =>varargs (exact match,promoting,wrapper)	

Private methods redeclaration

not inheritated
can redecclare 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

Java Class Design (OCA) Cheat Sheet

by Jianmin Feng (taotao) via cheatography.com/79308/cs/19394/

Inherite 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 overriden 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 overriden)
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 MIN_DEPTH=3	init at the statement
interface extends interface1,interface2,...	multiple extends allowed here
class inpluments interface1,interface2,...	multiple implements
can redecclare a method with same signature	
Rule for interface	
can not instantiated	may have no methods at all
public / default only	not private,final, protected for interface
all methods must be public	not private,final, protected for methods
abstract method by default	in java8, default, static methods with a body allowed
default interface methods	java 8
mainly for backward compatibility	
public default double calc(){}	

Interface (cont)

only in interface	can be redeclared as abstract or overriden with a different body
not static, final,or abstract (overriden)	not private,protected
Multiple inheritance problem	
default method in interface, if not overriden, will cause compiler error if default methods with same signuature existed	
for interfaces without default methods, there is no this issue	
if default method is overriden, also no ambiguity problem.	
Static method	java 8 above
public or default only	with a body
call with interface name, not with object ref	this avoid ambiguity cause be multiple inheritance
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. muliple inheritance of type:
https://docs.oracle.com/javase/tutorial/java/IandI/multipleinheritance.htm
https://www.baeldung.com/java-static-default-methods
https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-java/

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
overriden 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)

explicity 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 access directly. if by references, only by the ref of extending class itself

ref variables of other child class or event he parent class COULD NOT access parent members inside extending calss

constructor()

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

if class to be extends, 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 shared with all references.

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

variable on stack frame, obj on heap