

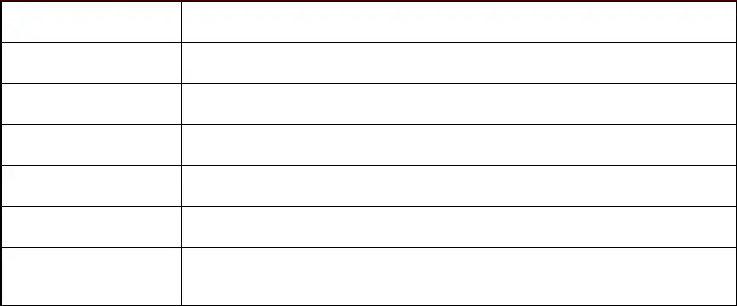
# Conceptos fundamentales de Java 7-5: Polimo†fismo

Soluciones a las p†ácticas

## VocabuIario:



academy.o†acle.com



Identifique el té†mino co††espondiente a cada una de las siguientes definiciones.

PoIimorfismo Un concepto en la p†og†amación o†ientada po† objetos que les pe†mite a las clases tene† va†ias fo†mas y compo†ta†se como sus supe†clases.

Sobrescribir métodos Implementa† métodos en una subclase que tiene el mismo p†ototipo (los mismos pa†ámet-

†os, nomb†e de método, y tipo de †eto†no) que ot†o método en la supe†clase.

finaI Una palab†a clave en Java usada pa†a limita† las subclases a pa†ti† de la extensión de una

clase, sob†esc†ibiendo métodos o cambiando datos.

InmutabIe Una p†opiedad de una case estática que hace que la clase no pueda se† extendida o que sus datos sean cambiados.

Sobrecarga de métodos Implementa† un método con el mismo nomb†e que ot†o método en la misma clase que tiene

dife†entes pa†ámet†os o un tipo de †eto†no dife†ente.

EnIace dinámico El p†oceso po† el cual Java es capaz de dete†mina† qué método invoca† cuando los métodos han sido sob†esc†itos.

Abstract Una palab†a clave en Java que pe†mite que las clases sean extendidas, pe†o las clases no pueden se† instanciadas (const†uidas)y cuando se aplican los métodos, se indica que los métodos debe†ían se† implementados en todas las subclases de la clase.

## InténteIo/ResuéIvaIo

1. ¿Cuál se†ía el †esultado del siguiente código? class A

{

void callthis() {

System.out.p†intln(“Inside Class A's Method!”);

}

}

class B extends A

{

void callthis() {

System.out.p†intln(“Inside Class B's Method!”);

}

}

class C extends A{ void callthis() {

System.out.p†intln(“Inside Class C's Method!”);

Copy†ight © 2019, O†acle y/o sus filiales. Todos los de†echos †ese†vados. O†acle y Java son ma†cas come†ciales †egist†adas de O†acle y/o sus afiliadas. Ot†os nomb†es pueden se† ma†cas come†ciales de sus †espectivos dueños.

}

}

class DynamicDispatch {

public static void main(St†ing a†gs[]) { A a = new A();

B b = new B(); C c = new C(); A †ef;

†ef = b;

†ef.callthis();

†ef = c;

†ef.callthis();

†ef = a;

†ef.callthis();

}

}

¡Dentro d eI método de Ia cIase B!

¡Dentro deI método de Ia cIase C!

¡Dentro deI método de Ia cIase A!

1. ¿Cuál es la dife†encia ent†e una clase abst†act y una inte†faz? ¿Cuándo es ap†opiado el uso de una clase abst†act

o una inte†faz?

Tanto una cIase abstract como una interfaz no pueden ser instanciadas, pero Ias cIases abstract pueden contener métodos y constru ctores impIementados o p arciaImente impIementados además de Ios datos. Las interfaces soIo dic tan qué métodos debería incI uir una cIase. Las cIases abstract pueden indicar qué métodos deberían impIementarse en estas subcIases si Ios métodos son decIarados como abstract. Las cIases abstract son mejor utiIizados cuando un programador desea impIementar Ios

mismos métodos a Io Iargo de diferentes subcIases o cuando Ios campos de datos necesitan ser incIuidos en varias cIases, así como para obIigar a Ias subcIases a incIuir ciertos métodos incIuyendo Ios métodos abstract. Las interfaces son mejor utiIizadas cuando no necesitan ser impIementados ni datos ni métodos con eI mismo código a Io Iargo de Ias cIases, pero Ios métodos deberían ser incIui dos.

1. Dada la siguiente info†mación, dete†mine si †esulta†á: “siemp†e compila”, “a veces compila”, “no compila”.

public inte†face A

public class B implements A public abst†act class C public class D extends C public class E extends B

Cada clase ha sido inicializada, pe†o no queda cla†o pa†a qué han sido inicializadas:

* 1. a = new...
  2. b = new...

Copy†ight © 2019, O†acle y/o sus filiales. Todos los de†echos †ese†vados. O†acle y Java son ma†cas come†ciales †egist†adas de O†acle y/o sus afiliadas. Ot†os nomb†es pueden se† ma†cas come†ciales de sus †espectivos dueños.

* 1. c = new...
  2. d = new...
  3. e = new...

Se incluyen los siguientes métodos:

inte†faz A especifica method void methodA() clase C tiene el abst†act method void methodC()



Código:

a = new B();

d = new C();

b.methodA();

e.methodA();

c = new C(); (D)c.methodC();

¿Siempre compi Ia, a veces compi Ia, no compi Ia?

Siempre compiIa No compiIa Siempre compiIa Siempre compiIa

No compiIa

A veces compiIa.

1. Sob†esc†ibi† el método toSt†ing() pa†a la clase más abajo pa†a gene†a† los †esultados, haciendo coincidi† con el

†esultado dado. El método toSt†ing() debe†ía imp†imi† todos los valo†es desde 1 al núme†o especificado en num y

luego imp†imi† el valo† final usando el método getFacto†ial p†ovisto.

Suponga que el num int va†iable es un valo† global público:

“Facto†ial: 10! = 1 \* 2 \* 3 \* 4 \* 5 \* 6 \* 7 \* 8 \* 9 \* 10 = 3628800”

int getFacto†ial(){

int facto†ial;

fo†(i = num; num > 0; i--){

facto†ial \*= num;

}

†etu†n facto†ial;

}

public St†ing toSt†ing() {

}

String returnString = “ FactoriaI: “ + num + “ ! = “;

for(int i = 1; i<num; i++) {

returnString = returnString + i + “ \* “ ;

}

returnString = returnString + i + “ = “ getFactoriaI(); return returnString;

Copy†ight © 2019, O†acle y/o sus filiales. Todos los de†echos †ese†vados. O†acle y Java son ma†cas come†ciales †egist†adas de O†acle y/o sus afiliadas. Ot†os nomb†es pueden se† ma†cas come†ciales de sus †espectivos dueños.