CENTRO UNIVERSITARIO UAEM ATLACOMULCO

MATERIA

PARADIGMAS DE LA PROGRAMACIÓN

DOCENTE

JULIO ALBERTO DE LA TEJA LÓPEZ

ALUMNO

ALEXIS VALENCIA MARTÍNEZ

CARRERA

LICENCIATURA EN INGENIERÍA EN SISTEMAS COMPUTACIONALES

ICO-27

FECHA DE ENTREGA

07/SEPTIEMBRE/2023

IMPLEMENTING FOR EXTENSIBILITY
With polymorphism, we can design and implement systems that are easily extensible [...]. For example, if we entend class ANIMAL to create class TORTOISE [...], we need to write only the TORTOISE class and the part of the simulation that instantiates a TORTOISE object. The portions of the simulation that tell each ANIMAL to move generically can ramain the same.

CHAPTER OVERVIEW
We then provide a simple example demonstrating polimorphic behaviour. We use superclass references to manipulate BOTH superclass object polymorphicall [...].

SPACE OBJECTS IN A VIDEOGAME
Suppose we design a video game that manipulates objects of classes Martian, Venusian, Plutonian, SpaceShip and LaserBeam. Imagine that each class inherits from the superclass SpaceObject, which contains method draw [...]. For example, a Martian object might draw itself in red with green eyes and the appropriate number of antennae. A SpaceShip object might draw itself as a bright silver flying saucer. A LaserBeam object might draw itself as a bright red beam across the screen [...]. The Java compiler does allow the assignment of a superclass reference to a subclass variable if we explicitly cast the superclass reference to the subclass type [...].
The program must first cast the superclass reference to a subclass reference through a technique known as downcasting. This enables the program to invoke subclass methods that are not in the superclass [...].

ABSTRACT CLASSES AND METHODS
When we think of a class, we assume that programs will create objects of that type. Sometimes it’s useful to declare classes-called abstract classes for which you never intend to create objects. Because they’re used only as superclasses in inheritance hierarchies, we refer to them as abstract superclasses [...].

DECLARING AN ABSTRACT CLASS AND ABSTRACT METHODS
Abstract methods do not provide implementations. A class that contains any abstract methods must be explicitly declared abstract even if that class contains some concrete (nonabstract) methods. Each concrete subclass of an abstract superclass also must provide concrete implementations of each of the superclass’s abstract methods. Constructors and static methods cannot be declared abstract. Constructors are not inherited, so an abstract constructor could never be implemented. Though non-private static methods are inherited, they cannot be overridden. Since abstract methods are meant to be overridden so that they can process objects based on their types, it would not make sense to declare a static method as abstract [...].

USING ABSTRACT CLASS TO DECLARE VARIABLES
Although we cannot instantiate objects of abstract superclasses, you’ll soon see that we can use abstract superclasses to declare variables that can hold references to objects of any concrete class derived from those abstract superclasses. We’ll use such variables to manipulate subclass objects polymorphically. You also can use abstract superclass names to invoke static methods declared in those abstract superclasses [...]. Consider another

application of polymorphism. A drawing program needs to display many shapes, including types of new shapes that you'll add to the system after writing the drawing program [...].

LAYERED SOFTWARE SYSTEMS
Polymorphism is particularly effective for implementing so-called layered software systems. In operating systems, for example, each type of physical device could operate quite differently from the others. Even so, commands to read or write data from and to devices may have a certain uniformity. For each device, the operating system uses a piece of software called a device driver to control all communication between the system and the device [...]. An object-oriented operating system might use an abstract superclass to provide an “interface” appropriate for all device drivers.

ABSTRACT SUPERCLASS EMPLOYEE
Each subclass overrides earnings with an appropriate implementation. To calculate an employee's earnings, the program assigns to a superclass Employee variable a reference to the employee's object, then invokes the earnings method on that variable. We maintain an array of Employee variables, each holding a reference to an Employee object. You cannot use class Employee directly to create Employee objects, because Employee is an abstract class [...].

FINAL METHODS AND CLASSES
A final method in a superclass cannot be overridden in a subclass this guarantees that the final method implementation will be used by all direct and indirect subclasses in the hierarchy. Methods that are declared private are implicitly final, because it's not possible to override them in a subclass. Methods that are declared static are also implicitly final [...].

FINAL CLASSES CANNOT BE SUPER CLASSES
A final class cannot be extended to create a subclass. All methods in a final class are implicitly final. Class String is an example of a final class. If you were allowed to create a subclass of String, objects of that subclass could be used wherever Strings are expected. Since class String cannot be extended, programs that use Strings can rely on the functionality of String objects as specified in the Java API. Making the class final also prevents programmers from creating subclasses that might bypass security restrictions [...].

STANDARDIZING INTERACTIONS
Interfaces define and standardize the ways in which things such as people and systems can interact with one another. For example, the controls on a radio serve as an interface between radio users and a radio's internal components [...]. The interface specifies what operations a radio must permit users to perform but does not specify how the operations are performed [...].

SOFTWARE OBJECTS COMMUNICATE VIA INTERFACES
A Java interface describes a set of methods that can be called on an object to tell it, for example, to perform some task or return some piece of information [...]. An interface declaration begins with the keyword interface and contains only constants and abstract methods. Unlike classes, all interface members must be public, and interfaces may not

specify any implementation details, such as concrete method declarations and instance variables. All methods declared in an interface are implicitly public abstract methods, and all fields are implicitly public, static and final [...]

STATICS INTERFACES METHODS

Prior to Java SE 8, it was common to associate with an interface a class containing static helper methods for working with objects that implemented the interface. For example, Collections method sort can sort objects of any class that implements interface List. With static interface methods, such helper methods can now be declared directly in interfaces rather than in separate classes [...].