OOP in Java – Paolo Perrotta

Approaching OOP

A diagram of a diagram

Description automatically generated

We say child is a subclass of parent

A screenshot of a computer

Description automatically generated

-We can pass in any subclass of Alarm

Functional Programming vs OOP

A black screen with white text

Description automatically generatedA screenshot of a black screen

Description automatically generated

-Primitives initialize by literal value

A black and white picture of objects

Description automatically generated

-Java still allocates memory when you create an object in Heap (reverse to objects)

-Returns a reference to object

-We prepare variable of the same type as the object, and assign the reference

Heap

-area memory where objects live

-like a big lake

A number and numbers on a white background

Description automatically generated with medium confidence

-Value is a Unicode sequence

A blue sign with white text

Description automatically generated

A computer screen with white text and green text

Description automatically generated

-Double =, comparing values (good for primitive) but Strings are objects (so need to compare reference) BUT IT MAY RETURN TRUE IN NEWER JAVA to spare memory

A blue sign with white text

Description automatically generated

Strings

-only objects without using new

-optimization to avoid creating too many strings

A screenshot of a cellphone

Description automatically generated

Final Object Variable

A black background with white text and orange and green text

Description automatically generated

-Only reference is final not object is not

A screenshot of a computer

Description automatically generated

-Strings are immutable, getting a new String BTS

AutoboxingA black background with white and orange text

Description automatically generated

-Integer is the wrapper class because you can wrap it around an int

-But Java does it automatically (Autoboxing) so you don’t have to do conversion

A black background with white text and blue numbers

Description automatically generated

-Looks like this behind the scene

A black background with white text and orange arrow

Description automatically generated

-Can even do this

Defining your own Classes

A blue background with white text

Description automatically generated

Unreachable Objects

-JVM can track references for us and if no object has no reference it will delete it for us (Garbage Collection)

-GC is unpredictable

-Java is not ideal for strict runtime computations

-No destructors in Java because of GC

A blue background with white text

Description automatically generated

-Like opening a file or network socket

Hiding Information

A screen shot of a computer code

Description automatically generated

-If you don’t declare a member public/private , that means the message is only visible to the same package

-Same thing applies to classes, must make it public to make it visible outside the package

A close-up of a chart

Description automatically generated

A white background with red text

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Designing with Abstraction and Encapsulation

A diagram of a computer code

Description automatically generated

-Design is translating the domain of the problem to the domain of the solution

A white and orange text on a white background

Description automatically generated

A white background with black text

Description automatically generatedInheriting from Another Class



A diagram of a computer

Description automatically generated

-use super.METHOD to use super methods

A diagram of a system

Description automatically generated

-If a class doesn’t explicit says extends, then they have implicit default superclass (Object from java.lang.package)

-All Java classes have ONE direct superclass

-toString() method comes from Object

A green and purple squares with white text

Description automatically generated

-final method can’t be overridden in a subclass

-final Class means you can’t inherit

Sealed Class

-it can decide exactly which other classes inherit from it

-Use cases are advanced



-Means only these classes can inherit it

-Subclasses must be sealed, non-sealed (Regular class) or final to remove errors

Understanding Polymorphism

Class Inheritance

-described as is-a relationship

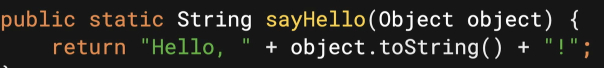
-Instance of a subclass IS A instance of superclass



-Subclass cannot make code less visible

A screenshot of a computer

Description automatically generated



-Can pass literally any Java Object because every Java Object is also an instance of Object

A screen shot of a computer

Description automatically generated

-Can only use Alarm methods

-Like breaking your TV remove and using a generic one

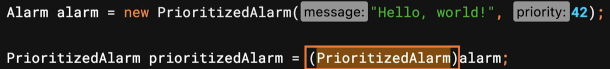
-So why when you lose some stuff?

A screen shot of a computer program

Description automatically generated

-It called the one from subclass, even though you cant access subclass, its still a subclass Object and will call subclass if overridden

Downcasting



Avoiding ClassCast expection

A green square with white text

Description automatically generated

Talking to Interfaces

Designing with Interitance and Polymorphism

Using the static Keyword

A blue sign with white text

Description automatically generated

Static Initializers aka Static block

-gets executed before calling constructor

A screen shot of a computer program

Description automatically generated

-don’t have to use MATH.PI



A blue sign with white text

Description automatically generated

Wrapping up

Enums, Records, Generic

A screen shot of a computer code

Description automatically generated