Cititi textul urmator (preluat de pe stackoverflow) si raspundeti la urmatoarele intrebari:

- *Ce doreste sa afle cel care a pus intrebarea?*
- *Ce a raspuns utilizatorul The Digital Gabeg? Sunteti de acord cu raspunsul sau?*
- *Ce comentarii a primit raspunsul? Au dreptate?*
- *Voi ce ati raspunde la intrebarea initiala?*

Polymorphism vs Overriding vs Overloading

In terms of Java, when someone asks:
what is polymorphism?
Would **overloading** or **overriding** be an acceptable answer?
I think there is a bit more to it than that.
IF you had a abstract base class that defined a method with no implementation, and you defined that method in the sub class, is that still overridding?
I think **overloading** is not the right answer for sure.

103
Polymorphism is the ability of a class instance to behave as if it were an instance of another class in its inheritance tree, most often one of its ancestor classes. For example, in Java all classes inherit from Object. Therefore, you can create a variable of type Object and assign to it an instance of any class.
An **override** is a type of function which occurs in a class which inherits from another class. An override function "replaces" a function inherited from the base class, but does so in such a way that it is called even when an instance of its class is pretending to be a different type through polymorphism. Referring to the previous example, you could define your own class and override the toString() function. Because this function is inherited from Object, it will still be available if you copy an instance of this class into an Object-type variable. Normally, if you call toString() on your class while it is pretending to be an Object, the version of toString which will actually fire is the one defined on Object itself. However, because the function is an override, the definition of toString() from your class is used even when the class instance's true type is hidden behind polymorphism.
Overloading is the action of defining multiple methods with the same name, but with different parameters. It is unrelated to either overriding or polymorphism.

The Digital Gabeg

- 9
  This is old but Polymorphism doesn't imply that the other class must be in the inheritance tree. It does in Java if you consider interfaces to be part of the inheritance tree, but not in Go, where interfaces are implemented implicitly.
  – J.N.
  Mar 1, 2011 at 14:51
- 5
  Actually, you don't need classes for polymorphism at all.
  – StCredZero
  Feb 1, 2013 at 22:39
- 4
  I'm a newbie, and correct me if I'm wrong, but I wouldn't say overloading is unrelated to polymorphism. At least in Java, polymorphism is when the implementation is chosen based on the type of the caller, and overloading is when the implementation is chosen based on the type of the parameters, isn't it? Seeing the similarity between the two helps me understand it.

– csjacobs24
Mar 26, 2015 at 20:13

- 14
  Incorrect. Ad hoc polymorphism is what you described in your **Overloading** section and *is* a case of polymorphism.
  – Jossie Calderon
  Jun 23, 2016 at 2:23
- 4
  "It is unrelated to either overriding or polymorphism". This statement is wrong.
  – Shailesh Pratapwar
  May 16, 2018 at 1:34