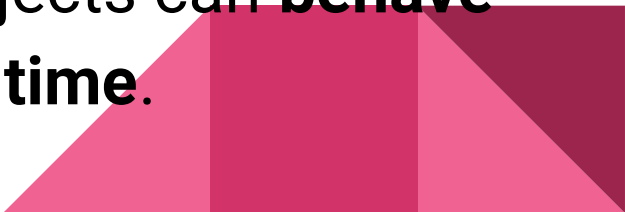


Polymorphism

Learning Objectives

- This lesson will explain that an object can have a different **declared (compile-time) type** than its **actual (run-time) type** depending on how the object is created.
 - Through inheritance (superclasses and subclasses), numerous different subclass objects can all be **declared as the type of the parent**. Then the objects can **behave as their specific subclass type at run-time**.
- 

Polymorphism

Polymorphism is a big word that you can break down into “**poly**” which means **many** and “**morphism**” which means **form**. So, it just means many forms.

In Java it means that **the method that gets called at run-time** (when the code is run) **depends on the type of the object at run-time.**

Polymorphism Analogy

This is similar to a toddler toy that has pictures of animals with a handle that when pulled causes an arrow spins. When the arrow stops the toy plays the sound associated with that animal.

If you were simulating this toy in software you could create an `Animal` class that had a `makeNoise` method. Each subclass of `Animal` would override the `makeNoise` method to make the correct noise for that type.

This type of polymorphism is called **inheritance-based polymorphism**. You have a common parent class, but the behavior is specified in the child class.



Variable Type

In Java an object variable has both a **declared (compile-time) type** and an **actual (run-time) type**. The declared (compile-time) type of a variable is the type that is used in the declaration. The actual (run-time) type is the class that actually creates the object using new.

Declared type of
`nameList` is
`List`



Run-time type of
`nameList` is
`ArrayList`



```
List<String> nameList = new ArrayList<String>();  
nameList.add("Hi");
```

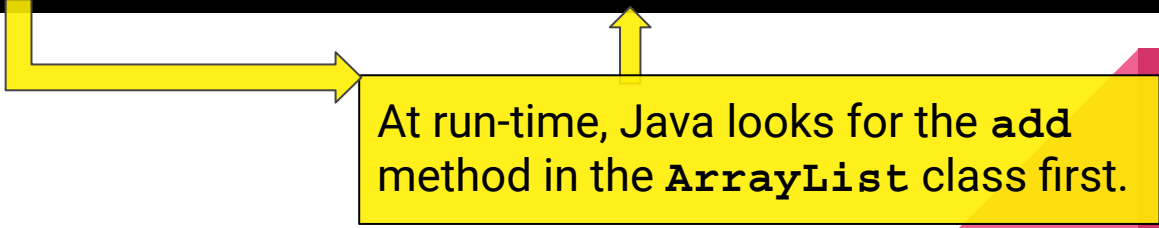
Run-time progression

At run-time, the execution environment will first look for the `add` method in the `ArrayList` class since that is the actual or run-time type.

If it doesn't find it there it will look in the parent class and keep looking up the inheritance tree until it finds the method.

It may go up all the way to the `Object` class. The method will be found, since otherwise the code would not have compiled.

```
List<String> nameList = new ArrayList<String>();  
nameList.add("Hi");
```



At run-time, Java looks for the `add` method in the `ArrayList` class first.

Compile time, Run time

At compile time, the compiler uses the declared type to check that the methods you are trying to use are available to an object of that type.

The code won't compile if the methods don't exist in that class or some parent class of that class.

At run-time, the actual method that is called depends on the actual type of the object. When a method is called at run-time the first place that is checked for that method is the class that created the object. If the method is found there it will be executed. If not, the parent of that class will be checked and so on until the method is found.

