Inheritance Basics

S. Ducasse, L. Fabresse, G. Polito, and P. Tesone

http://www.pharo.org
Goal

- What is inheritance?
- When to use it?
- BTW, Pharo has the same inheritance model as Java
Inheritance

- It is a reuse mechanism
  - We do not reimplement the code of the superclasses
  - We extend it or customize it
- It is based on the expression of a delta
  - Specify only the differences to the superclasses
The basics

Needs:
- We want to adapt the code by extending existing behavior and state
- We do not want to reimplement everything

Solution: **class inheritance**
- A class extends the definition of its superclass
Basic subclass behavior

A subclass:

- can **add** state and behavior: color, borderColor, ...
- can **use** superclass behavior and state
- can **redefine** superclass’ behavior to specialize it

```
Rectangle
| width |
| height |
| area() |
| ... |

ColoredRectangle
| color |
| borderColor |
| color() |
| ... |
```
Root of inheritance hierarchy

- **Object** is the root of most classes
  - defines the common behavior of all objects
  - raising errors, class access, ...

```
<table>
<thead>
<tr>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>doesNotUnderstand()</td>
</tr>
<tr>
<td>class()</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
</tr>
<tr>
<td>height</td>
</tr>
<tr>
<td>area()</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
```
ProtoObject (Object’s superclass) has a special purpose:

- raising as much errors as possible
- so that the system can catch such errors and do something with them
- useful for building advanced techniques such as proxy objects
Two aspects of inheritance

Inheritance is:

- **static** for state/instance variables (i.e. during class creation)
- **dynamic** for behavior (i.e. during execution)
Inheritance of instance variables

- Happens during **class definition**
- Computed from
  - the class own instance variables
  - the ones of its superclasses
  - usually no duplicate in the chain
- ColoredRectangle **has a width, height, color, and borderColor**
Inheritance of behavior

- Happens at **run time**
- The method is looked up
  - starting from the receiver’s class
  - then going along superclasses
What you should know

- Inheritance allows developers of a class to add state and behavior and redefine behavior.
- A class has 1 and only 1 superclass (single inheritance model).
- A class eventually inherits from Object.
- Inheritance of state is static.
- Inheritance of behavior is dynamic.