UML - Object Diagrams

Linda Marshall and Vreda Pieterse

Department of Computer Science
University of Pretoria

2014

Overview

Object diagrams:

- are derived from **Class diagrams** and are therefore dependent on class diagrams
- represent an instance of a class diagram
  - also a static view
  - a snapshot of the system at a specific moment
- are used for forward and reverse engineering

Object diagrams (cont):

- are concrete in nature, represent the real-world **vs** class diagrams that are abstract and represent the blue-print
- have unlimited instances **vs** fixed classes of class diagrams
- use the same basic relationships to class diagrams

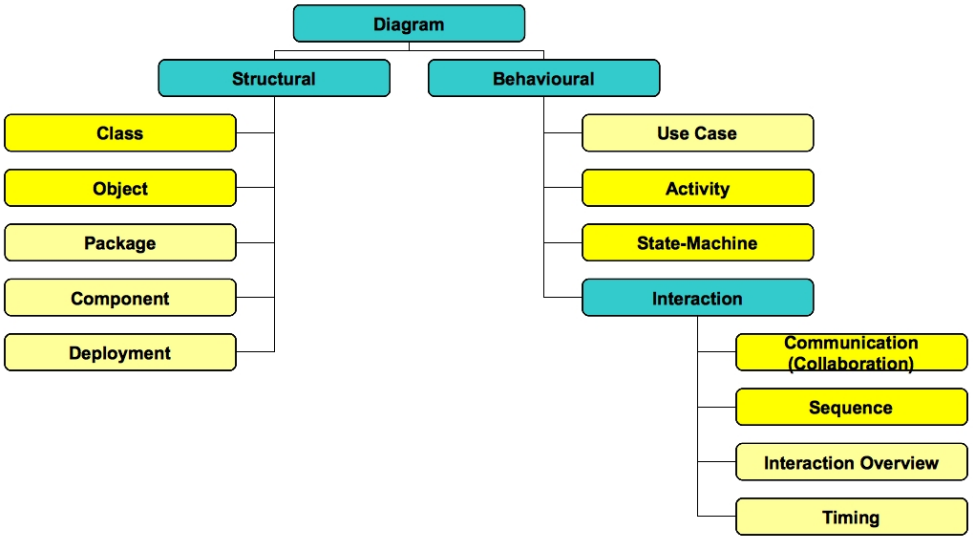
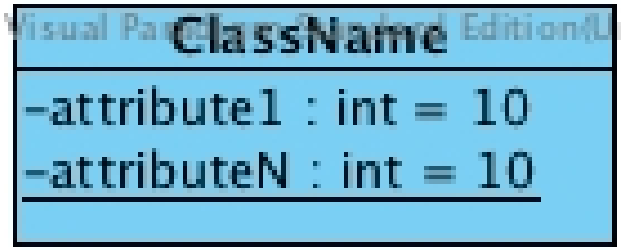
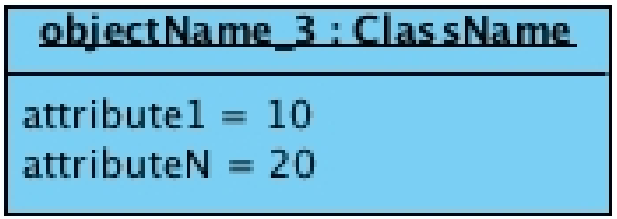


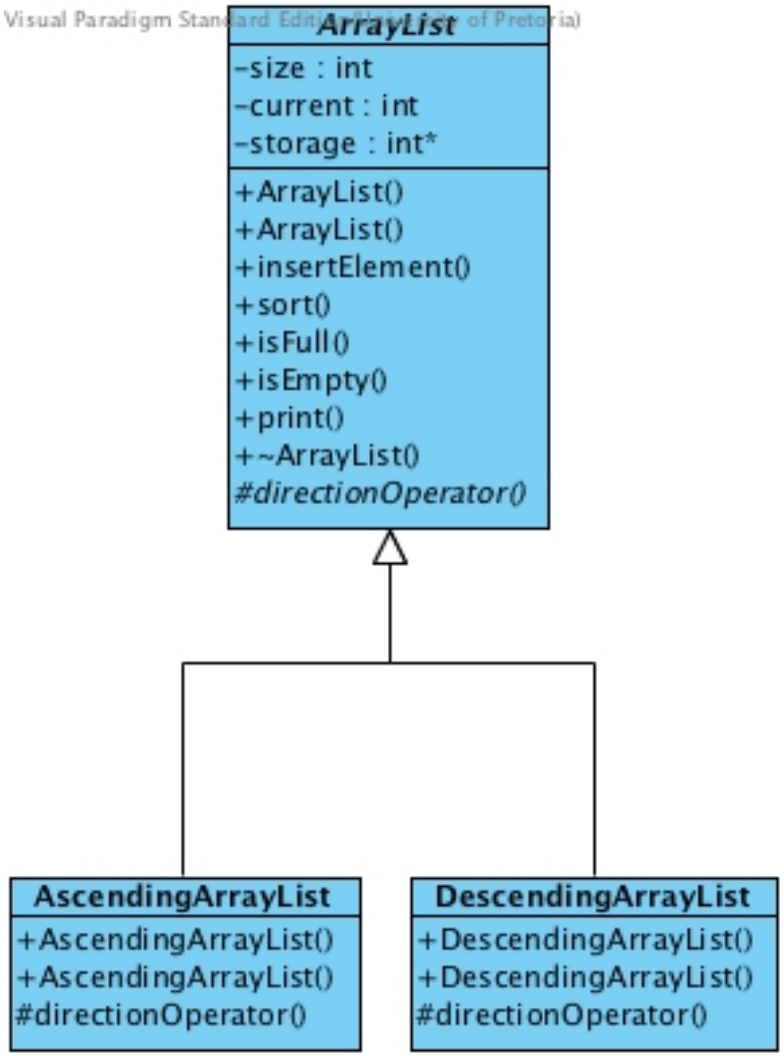
Diagram
Structural
Behavioural
Class
Object
Package
Component
Deployment
Use Case
Activity
State-Machine
Interaction
Communication (Collaboration)
Sequence
Interaction Overview
Timing


objectName_1


objectName_2 : ClassName

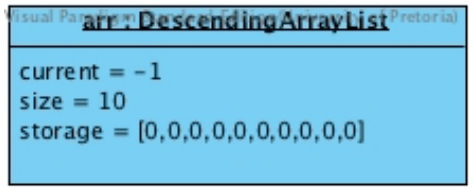

ClassName
-attribute1 : int = 10
-attributeN : int = 10


objectName_3 : ClassName
attribute1 = 10
attributeN = 20

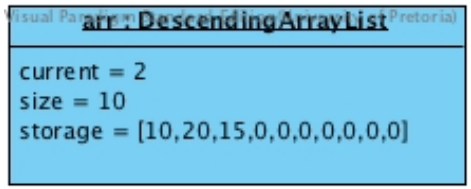

Visual Paradigm Standard Edition
of Pretoria)
ArrayList
-size : int
-current : int
-storage : int*
+ArrayList()
+ArrayList()
+insertElement()
+sort()
+isFull()
+isEmpty()
+print()
+~ArrayList()
#directionOperator()
AscendingArrayList
+AscendingArrayList()
+AscendingArrayList()
#directionOperator()
DescendingArrayList
+DescendingArrayList()
+DescendingArrayList()
#directionOperator()

```
1   ArrayList* arr = new DescendingArrayList(10);
2
3   arr->insertElement(10);
4   arr->insertElement(20);
5   arr->insertElement(15);
6   arr->insertElement(25);
7   arr->insertElement(5);
8
9   arr->print();
10  arr->sort();
11  arr->print();
```

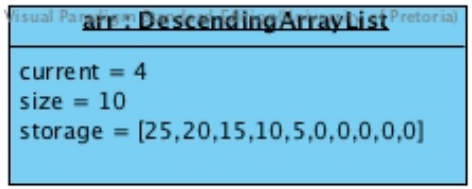
Line 1

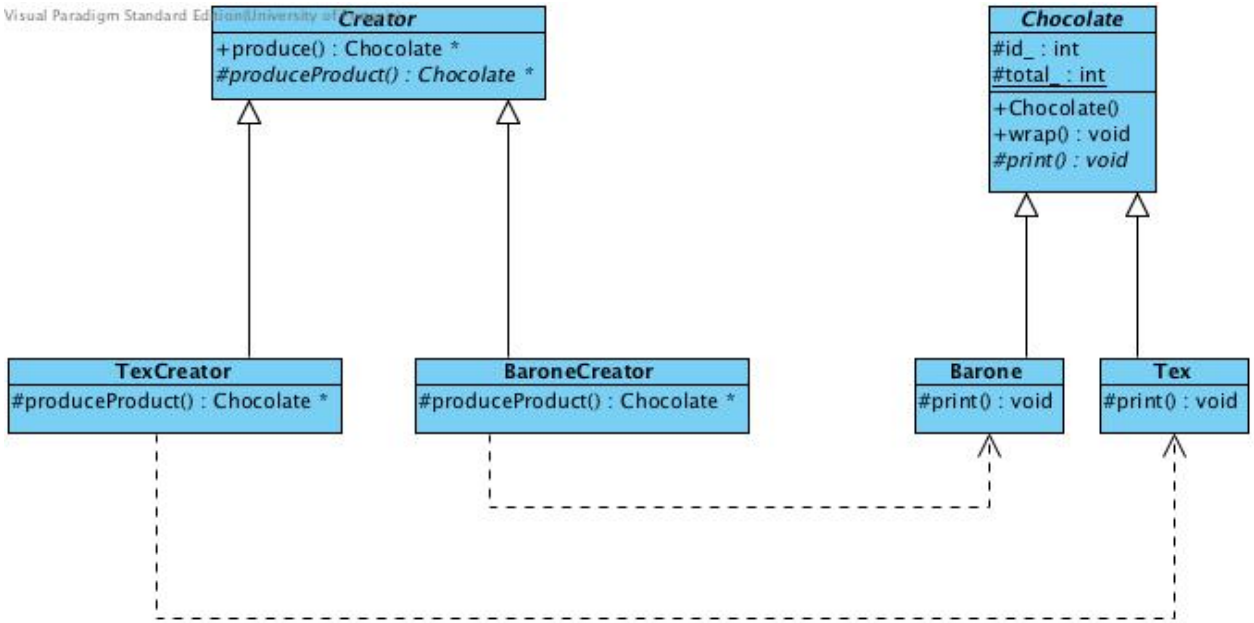


Line 5



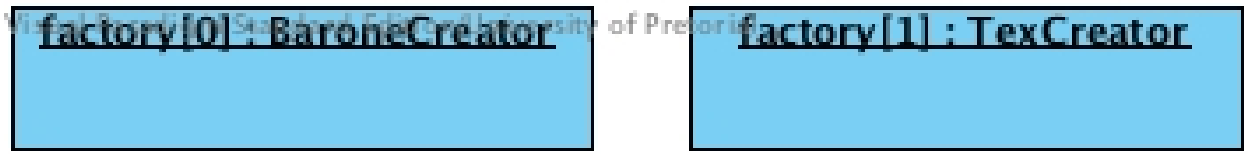
Line 10



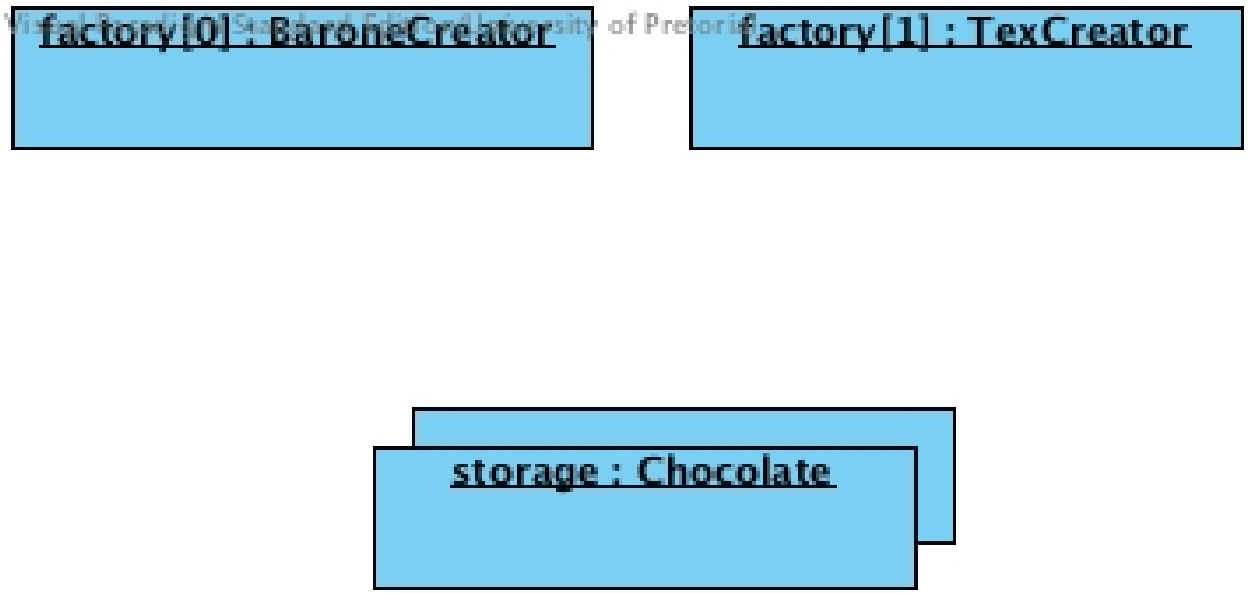

Creator
+produce() : Chocolate *
#produceProduct() : Chocolate *
Chocolate
#id_ : int
#total_ : int
+Chocolate()
+wrap() : void
#print() : void
TexCreator
#produceProduct() : Chocolate *
BaroneCreator
#produceProduct() : Chocolate *
Barone
#print() : void
Tex
#print() : void

```
Creator* factory[2];
```

```
factory[0] = new BaroneCreator();
factory[1] = new TexCreator();
```



```
Chocolate* storage[5];
```



for $i < 2$:

```
for (int i = 0; i < 5; i++)
  storage[i] =
     factory[i%2]->produce();
```