Finish Case Study, More Design Patterns

CS151

Chris Pollett

Nov. 14, 2005.

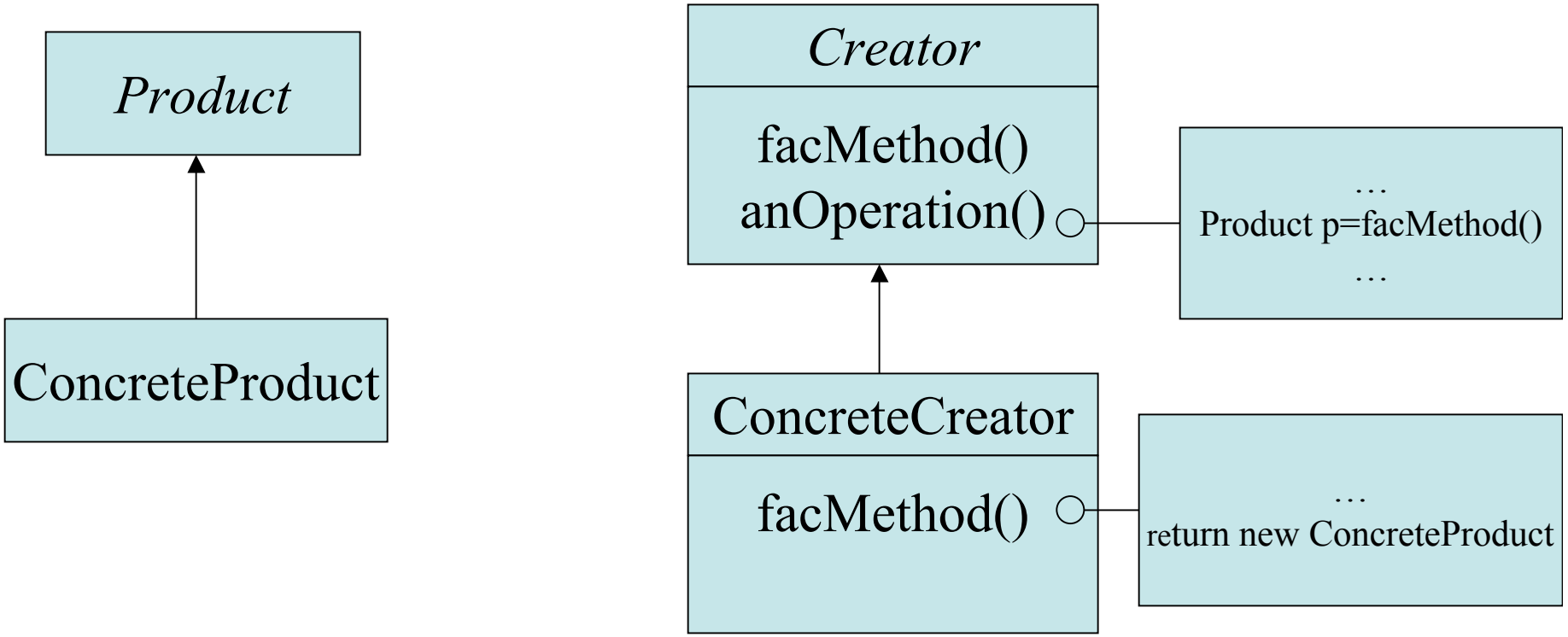
Outline

- Iterations 4-6
- Enumeration Types

Iteration 4

- We almost finished talking about Iteration4 before the midterm.
- One last thing about Iteration 4 is that it makes use of the creational design pattern known as the Factory method pattern.
- For instance, the class DrawingPad extends the class Scribble.
- The class Scribble has a makeCanvas() method used to make a ScribbleCanvas.
- In DrawingPad, this method is overriden to create a DrawingCanvas which extends a ScribbleCanvas and has additional methods for getting and setting the tool.

Factory Methods



Iteration 5

- This iteration adds tools for ovals and rectangles
- A new subclass of Shape TwoEndsShape is created and a new tool TwoEndTool is created which extends Tool to support it.

Iteration 6

- In this iteration another type of Tool called a KeyboardTool is created with a method addCharToShape. It is subclassed to make a TextTool to support text entry is created.
- Similarly, another subclass of Shape called Text is added.
- The relevant Events and Listener for these extensions are KeyEvent and KeyListener.

Maze Game

- In Chapter 10 a maze game is developed.
- The board consists of a n x m grid. Each square on the board is a room.
- Adjacent rooms are separated by either a wall or a door.
- Doors can be open or closed.
- A player can move around rooms.
- To represent this game it is useful to be able to say if a door is to be represented horizontally or vertical and it is useful to be able to say the directions NORTH, SOUTH, EAST, WEST.

Enumeration Types

- One could spell out these types using:

```
public interface Orientation
{
    public static final int HORIZONTAL = 0;
    public static final VERTICAL =1;
}
public interface Direction
{
    public static final int NORTH= 0;
    public static final int EAST= 1;
    public static final SOUTH =2;
    public static final int WEST= 0;
}
```

- However, it is not type safe.

Java 1.5 Enumerated Types

```
public enum Orientation
{
   HORIZONTAL, VERTICAL
}
```

- This generates a class Orientation with static variables HORIZONTAL, and VERTICAL.
- This class also has methods values(), valueOf(). For example, can do Orientation o = Orientation.valueOf(“VERTICAL”);
- Equals, etc are overload appropriately.

How could we do something like this in Java 1.4?

- The book suggests the following type-safe enumeration idiom:

```
public class Orientation
{
    public static final Orientation VERTICAL = new
      Orientation("VERTICAL");
    public static final Orientation HORIZONTAL = new
      Orientation("HORIZONTAL");
    public String toString() { return name;}
    private Orientation(String name){this.name = name;}
    private final String name;
}
```

Ordered Type-Safe Enumeration

- How can we get a type-safe enumeration where we have an ordering on the elements of that type.

```
public class Direction implements Comparable
{
    public static final Direction NORTH = new Direction("North");
    //other directions
    //toString,
    public int getOrdinal(){ return ordinal;}
    public int compareTo(Object o) { if (o instanceof Direction) {return ordinal -
        ((Direction )o).getOrdinal()}
    }
    // first, next, opposite,etc

}
```

More on the Maze Game

- As a first pass of the maze games we have a class Maze.
- This has on it one or more Room's.
- A Room is a subclass of a MapSite interface which supports the methods clone(), enter(Maze maze), draw.
- Other subclasses include Wall and Door.
- A driver application SimpleMazeGame creates a maze and all of the objects on it.