CMSC 132: OBJECT-ORIENTED PROGRAMMING II

Singleton and Decorator Design Patterns

Department of Computer Science
University of Maryland, College Park

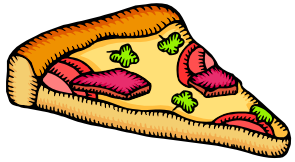
Singleton Pattern

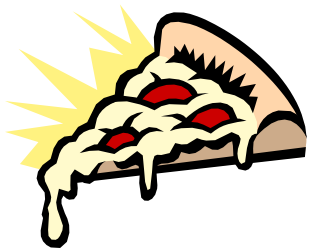
- **Definition**
  - One instance of a class or value accessible globally
- **Where to use & benefits**
  - Ensure unique instance by defining class final
  - Access to the instance only via methods provided
- **Example**

```
public class Employee {
    public static final int ID = 1234;              // ID is a singleton
}

public final class MySingleton {

    // declare the unique instance of the class
    private static MySingleton uniq = new MySingleton();

    // private constructor only accessed from this class
    private MySingleton() { … }

    // return reference to unique instance of class
    public static MySingleton getInstance() {
        return uniq;
    }
}
```

Decorator Pattern

- **Definition**
  - Attach additional responsibilities or functions to an object dynamically or statically
- **Where to use & benefits**
  - Provide flexible alternative to subclassing
  - Add new function to an object without affecting other objects
  - Make responsibilities easily added and removed dynamically & transparently to the object
- **Example**
  - Pizza Decorator adds toppings to Pizza
  - Original
    - Pizza subclasses
    - Combinatorial explosion in # of subclasses
  - Using pattern
    - Pizza decorator classes add toppings to Pizza objects dynamically
    - Can create different combinations of toppings without modifying Pizza class
    - **Example:** PizzaDecoratorCode





Decorator Pattern

- Examples from Java I/O
  - Interface
    - InputStream
  - Concrete subclasses
    - FileInputStream, ByteArrayInputStream
  - Decorators
    - BufferedInputStream, DataInputStream
  - Code
    - InputStream s = new DataInputStream(new BufferedInputStream (new FileInputStream()));