jpatterns.org

Dr Heinz M. Kabutz
heinz@javaspecialists.eu
http://www.javaspecialists.eu

Javaspecialists.eu
java training

Brief Biography

- Dr Heinz Kabutz
  - Live on Island of Crete in Mediterranean Sea (Greece)
  - PhD Computer Science from University of Cape Town
  - The Java Specialists' Newsletter
  - Java programmer
  - Java Champion since 2005
  - Java instructor to corporates
    - Java Patterns Course
    - Java Specialist Master Course
      - Threads, Java NIO, Memory, Optimizations, etc.
      - Requires 2 years solid Java experience to participate
      - Chania (Crete) May 2011 & September 2011
      - Remote evening classes in January

Why Crete?

- **Airport 10 minutes from my house**
- **E1 connection to my house**
- **Closer to customers than Cape Town**
- **Great lifestyle, good food, clean air**
- **Super friendly citizens**
- **Wife and children are Greek citizens**
- **And now for the *real reason* ...**

Java Specialist Club

- Fitness club for the mind
- Learning webinars
  - Design Patterns
  - Java
  - Consulting profession
- Forum Discussion
- Seeding open source projects
- www.javaspecialists.eu/club

Who is involved with jpatterns.org?

- **Project leaders**
  - **Michael Hunger**
  - **Heinz Kabutz**
- **Project participants**
  - **Marco Tedone**
  - **Johannes Bühler**
  - **Alex Gout**
- **All are welcome to join and help**

What is jpatterns.org?

- **Java annotations for describing patterns in code**
- **Formalises pattern usage**
- **In future, we might write tools to extract annotations to help describe systems**

What is jpatterns.org *not*?

- **A set of tools for verifying correct implementation of patterns**
  - **Patterns help us get started, but they are not final solution**
  - **The structure is the weakest dominator in the pattern**
    - **More important is intent and name**
    - **jpatterns annotates your intent**
  - **How would you verify that a class is a Singleton?**

Example Adapter Without Annotations

```
public class Rapper {
  public String talk() {
    return "Vulgar lyrics deleted...";
  }
}

public class RapperClassAdapter
    extends Rapper implements Singer {
  public String sing() {
    return talk();
  }
}
```

Example Adapter With Basic Annotation

```
public class Rapper {
  public String talk() {
    return "Vulgar lyrics deleted...";
  }
}

import org.jpatterns.gof.*;

@AdapterPattern
public class RapperClassAdapter
    extends Rapper implements Singer {
  public String sing() {
    return talk();
  }
}
```

Example Adapter With Detailed Annotation

```
@AdapterPattern.Adaptee
public class Rapper {
  public String talk() {
    return "Vulgar lyrics deleted...";
  }
}

@AdapterPattern.Adapter(
    value = AdapterPattern.Variation.CLASS,
    participants = {Rapper.class, Singer.class})
public class RapperClassAdapter
    extends Rapper implements Singer {
  public String sing() {
    return talk();
  }
}
```

Where can I find the annotations?

- Our jpatterns-0.1.jar file is available here
  - http://www.jpatterns.org/download
- Javadocs are here:
  - http://www.jpatterns.org/javadocs
- For more information, look at
  - http://www.jpatterns.org

Why do we need this?

- Programmers design using well established patterns
- The pattern might not be that obvious to others
- e.g. JUnit was developed test-driven, but Gamma and Beck were talking in patterns

Classic Methodologies

- **e.g. Waterfall Model: Analysis, Design, Implementation, Testing**
- **Suffers from “Analysis Paralysis”**
- **Bad decision during analysis very expensive**
- **Model for large teams with greatly varying skill-sets**
- **Each iteration takes months**

Agile Methodologies

- e.g. eXtreme Programming
- All programming is done in pairs
  - For constant code reviewing, NOT mentoring
- Very short iterations (days or weeks)
- Testing is done several times a day
- Daily automated build and complete test
- Designing with Patterns
- Ruthless refactoring

Which Methodology to Use?

- **Waterfall Model**
  - One or two excellent analysts
  - Few good designers
  - Lots of average programmers
  - Suffers from “Peter Principle”
- **eXtreme Programming**
  - Between 6 and 12 above average programmers per team
  - Fosters cooperation, not competition in team
  - Low staff turnover
  - Chaos if not strictly managed!!!

Typical Day as Programmer

- Let's look at Joe's day at work:
  - 08:00 Arrive at work
  - 08:30 Had first cup of coffee, erased SPAM
  - 09:00 Chatted with coworker about soccer
  - 10:00 Had project status meeting
  - 11:00 Thought about design problems
    - (Whilst playing minesweeper)
  - 12:30 Looked at some critical bugs for important customer
  - 13:30 Finished playing “Battlefield 1942” with colleagues
  - 15:00 Wrote 200 lines of VB code, answered 5 phone calls
  - 16:30 Company meeting entitled “Be more productive”
  - 17:30 Wrote emails to bosses and colleagues (and friends)
  - 23:30 Time to go home – finished writing TCP/IP stack in assembler

Programming is a Minority Task

- **Most of your time is spent in:**
  - **Meetings**
  - **Documentation**
  - **Planning**
  - **Testing, bug fixing & support**
  - **Email**
- **Even brilliant programmers have to communicate!**

Design Language can Help

- Meetings
  - Communicate more effectively about your designs to colleagues
- Documentation
  - Code documentation can refer to Design Pattern
- Planning
  - You can talk in higher-level components
- Testing, bug fixing & support
  - Better designs will reduce bugs and make code easier to change

Organic First Cold Pressed Virgin Olive Oil

- **Design Patterns are like good olive oil**
  - **You cannot appreciate them at first**
  - **As you study them you learn the difference between supermarket oil and the real stuff from Heinz's farm**
  - **As you become a connoisseur you experience the various textures you didn't notice before**
- **Warning: Once you are hooked, you will no longer be happy with bottled oil!**

Design Patterns Origin

The Timeless Way of Building

Christopher Alexander

There is a central quality which is the root criterion of life and spirit in a man, a town, a building, or a wilderness.

If you want to make a living flower, you don't build it physically, with tweezers, cell by cell. You grow it from the seed.

Why are patterns so important?

- Provide a view into the brains of OO experts
- Help you understand existing designs
- Patterns in Java, Volume 1, Mark Grand writes
  - "What makes a bright, experienced programmer much more productive than a bright, but inexperienced, programmer is experience."

What's in a name?

The Timeless Way of Building

The search for a name is a fundamental part of the process of inventing or discovering a pattern.

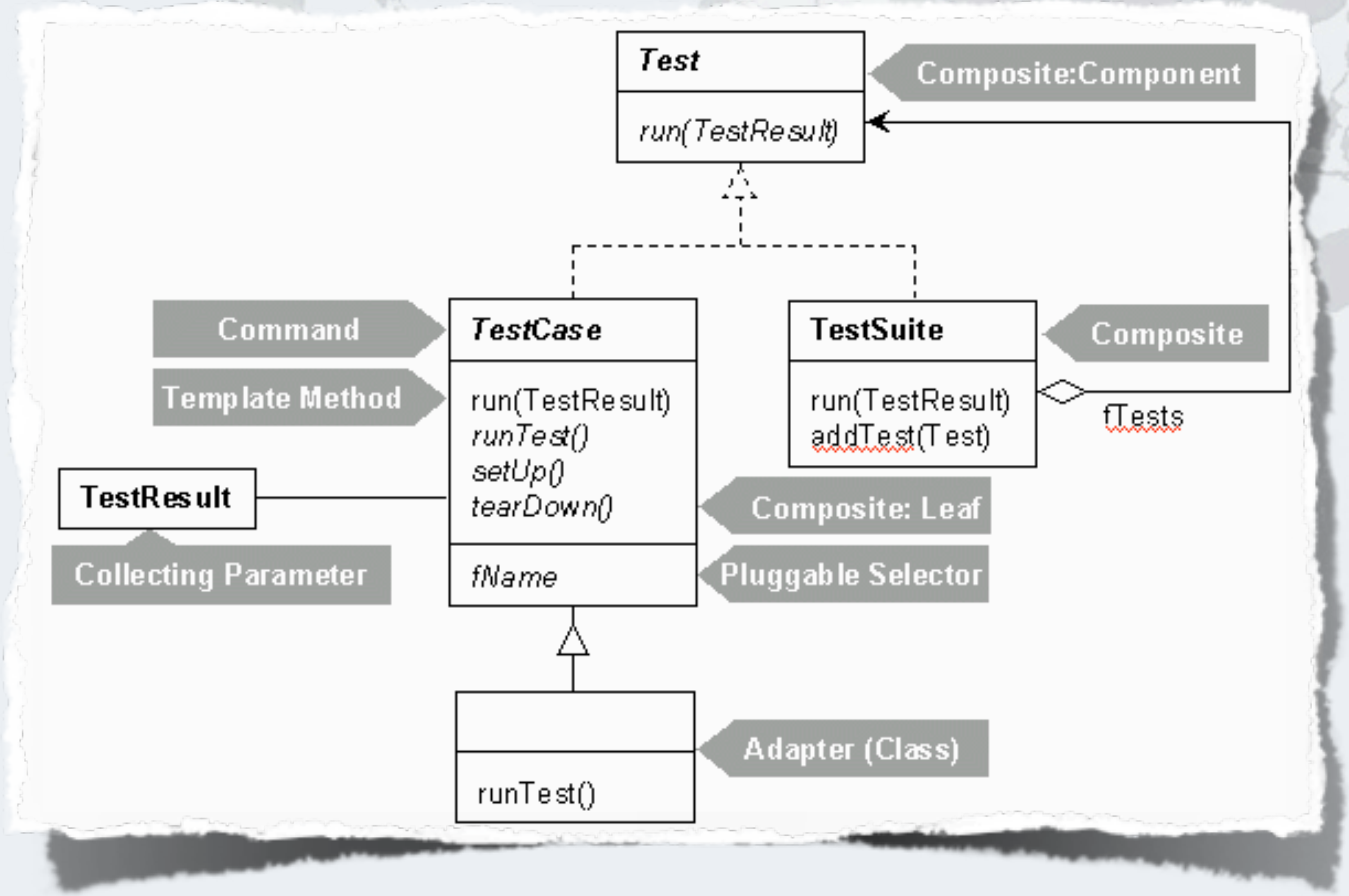
So long as a pattern has a weak name, it means that it is not a clear concept, and you cannot tell me to make "one".

Where are patterns usually documented?

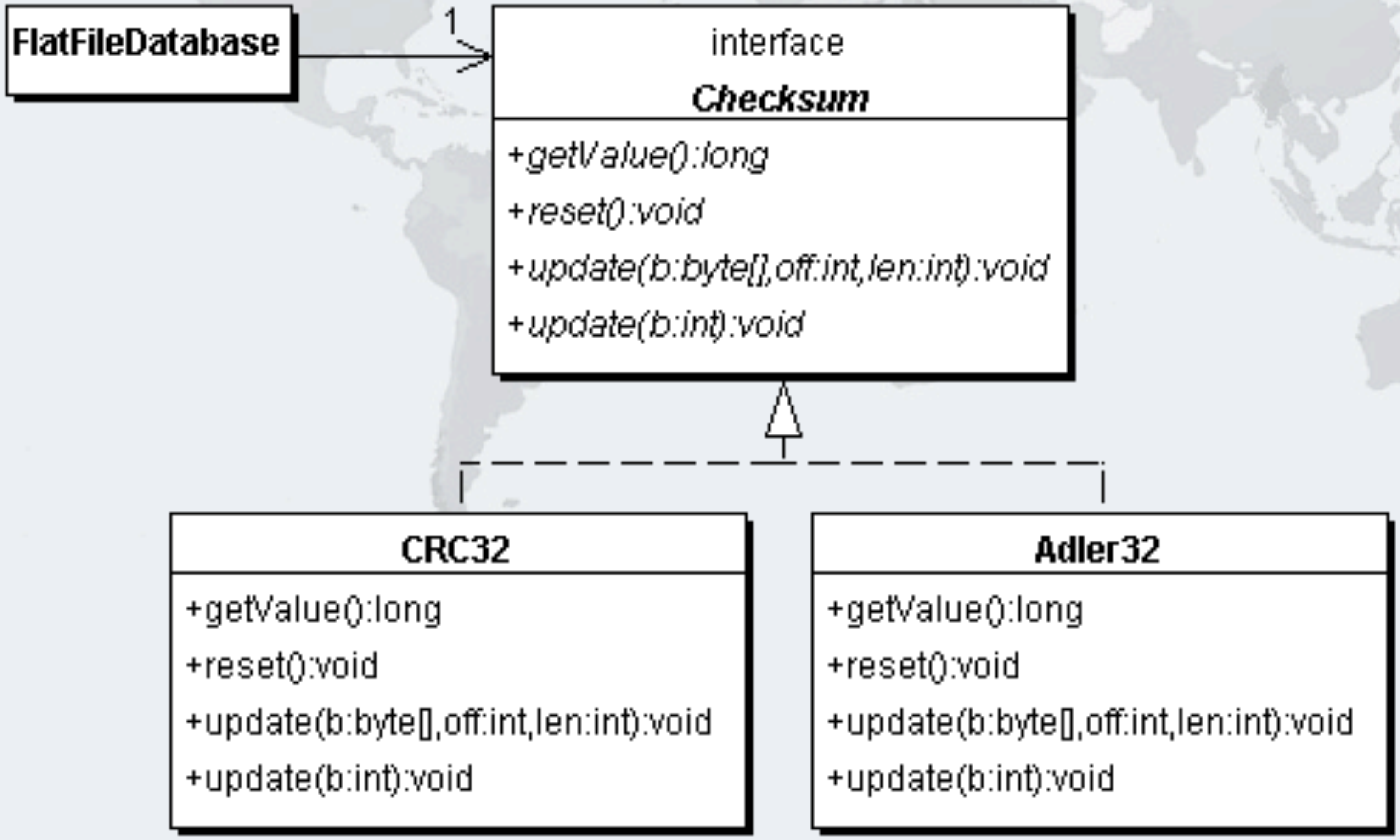
- In the class name: TreeVisitor, FilteredOutputStream
  - But often not, e.g. Runtime, Runnable, Checksum
- In UML class diagrams as text notes
  - But seriously, who draws UML diagrams nowadays?
  - There is no proper tooling support
- In the JavaDocs, but inconsistently
- In a separate design document
  - http://junit.sourceforge.net/doc/cookstour/cookstour.htm

JUnit Patterns Overview

- **From the JUnit Cook's Tour**



How do the annotations work?



JPatterns.org Annotations

```
@StrategyPattern.Strategy
public interface Checksum {
  long getValue();
  void reset();
  void update(int b);
}

@StrategyPattern.ConcreteStrategy
public class Adler32 implements Checksum {
  ...
}

@StrategyPattern.Context
public class FlatFileDatabase {
  @StrategyPattern.StrategyField
  private final Checksum checksum;
}
```

Demo

Annotating junit

Questions?

Javaspecialists.eu
java training

jpatterns.org

Dr Heinz M. Kabutz

heinz@javaspecialists.eu

http://www.javaspecialists.eu

Javaspecialists.eu
java training