Aalto University
School of Science
and Technology

Methodology for Computer Science Research
Lecture 5: Academic Programming

Andrey Lukyanenko

Department of Computer Science and Engineering
Aalto University, School of Science and Technology
andrey.lukyanenko@aalto.fi

October 18, 2012

- What is academic programming?
- What tools and languages to use?
- Development models
- Design patterns and pitfalls
- Task

On the paper.

Do you remember paper outline?

Section	Content
-	Title
-	Abstract
1	Introduction
2	History (Related work)
3	Idea, Algorithm
4	Model
5	Simulation, Measurements
6	Evaluation, Data Analysis
7	Implementation (Demo)
8	Discussion (Results), Future work
9	Conclusion
-	Reference

Life-cycle of MS thesis?

Normal process for MS thesis development consists of:

1. Selection of a topic.
2. Talking to the topic "owner".
3. Adjusting the topic, if the student has a background in the topic.
4. Reading and studying (one month).
5. Developing a code (one month).
6. Testing, analyzing, improving, result collecting (one month).
7. Writing the thesis (at fastest average is one page per day, total >50).

In total: 6 month with fast pace.

Some differences in PhD studies

- Normally, PhD students have one year for initial study. During that time they also select a topic.
- Big difference in working plan in different countries, for example:
  1. In Finland:
     - PhD consists of many iterations like that in MS thesis preparation process.
     - One or pair of iterations result in an article.
     - At the end, a PhD student should have many small conference papers (possible incremental work for a journal article).
  2. In USA:
     - PhD is to write one big article: a lot of time for studying the field and a lot of time (as well as iterations) to write the actual code.
     - In the end, student will have a well written code, as well as one big well written article.
     - This big article produces some spin-offs and smaller papers, but one big article is enough to receive a PhD degree.

What is Academic programming?

To do thesis you need to do “Academic programming”. The main differences compared to commercial programming as well as features:

- Develop your application very fast.
- One person team (develop alone).
- Make the program without specification and clients.
- No unit tests.
- Expect “turbulence” in the programming (sudden change of what you need to do).
- You focus on back-end (business logic), instead of frond-end (user interface).
- Very small iterations between introducing new feature and testing it.
- Showing intermediate result to instructor.

Academic programming: steps

To defend yourselves from unexpected changes, you need

- Smartly choose the tool;
- Study what you need to do before programming;
- Study how others do it;
- Use best practices;
- Make a plan, with short deadlines;
- Talk to your instructor each week on progress you have done and progress you will do;
- Document the progress and all the milestones (for example, in thesis draft document).

- What is academic programming?
- **What tools and languages to use?**
- Development models
- Design patterns and pitfalls
- Task

Choosing the tool for programming

When you select a tool for development you should consider the following:

- `[Always]` Advisor says that you need to implement a specific feature.
- `[Often]` Advisor mentions a system for which it is needed.
- `[Seldom]` Advisor mentions a tool which to use.
- `[Often]` You need to find the tool yourself.
- `[Rarely]` You have no option in selecting the tool.
- `[Very rarely]` Advisor explicitly tells what and how to program.

Tools and languages

Often used:

Java, Python, C++, C#, MatLab, R and etc.

Name	IDE	Use for
Java	Eclipse	Windows/Linux/Android/Blackberry
Python	Eclipse	AppEngine/Web/NS-3
C	Eclipse CDT/Visual Studio	Linux Kernel
C++	Eclipse CDT/Visual Studio	Windows/Linux/NS-3/OMNeT++
C#	Visual Studio/MonoDevelop	Windows/Windows Phone/Mono
MatLab	MatLab	Mathematical programming
R	Any editor	Graphics/Statistics
PHP	Eclipse PDT/Aptana	Web development together with MySQL
JavaScript	Aptana	Dynamic HTML development
Excel	MS Word/LibreOffice Calc	Statistics
Object C	Mac XCode	iPhone

Smart choice of a tool

Sometimes few languages and many tools exist for your project. Conventionally people choose a tool which is

- free (students do not expect to pay for the SW).
- popular (as a proof-of-usability).
- well-documented.
- aware of the language you know (or willing to learn).

Some software can be obtained through the university access. At Aalto students have an access to `download.aalto.fi` (MatLab?) and MS DreamSpark program (free Visual Studio).

An example: Tools for simulators

Assume you need to simulate some mechanism or protocol (quite a common use case).

- Simulate or emulate? (Defines a precision with which a simulator should work)
- How important is speed? (Complexity of the individual task).
- Generic or narrow purpose?
- How important is the language? C++ a common choice, Java the next.

The above decisions produce one of the following results. Use OMNeT++, NS2/NS3, Own Simulator, Real-life code which can be added to simulator or real implementation (e.g., OpenFlow, Click modular router, PlanetLab).

- What is academic programming?
- What tools and languages to use?
- **Development models**
- Design patterns and pitfalls
- Task

Development models

For commercial software there are different development methods/models:

- Waterfall model.
- Spiral model.
- Iterative and incremental development
- Agile development

All of them are too “slow” for Academic programming. The closest is Agile SW development, which in turn consists of

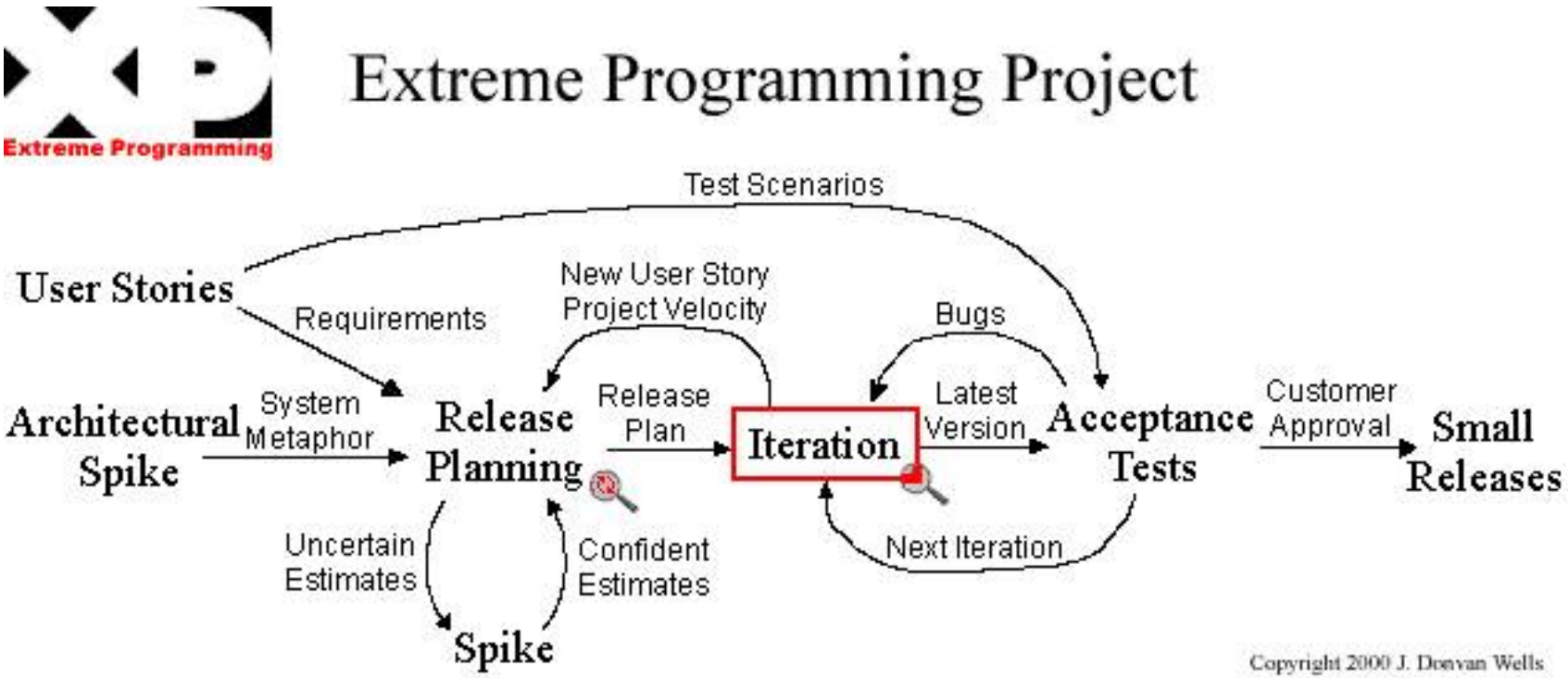
- Extreme Programming (XP).
- Scrum.
- etc.

Extreme Programming

XP main features:

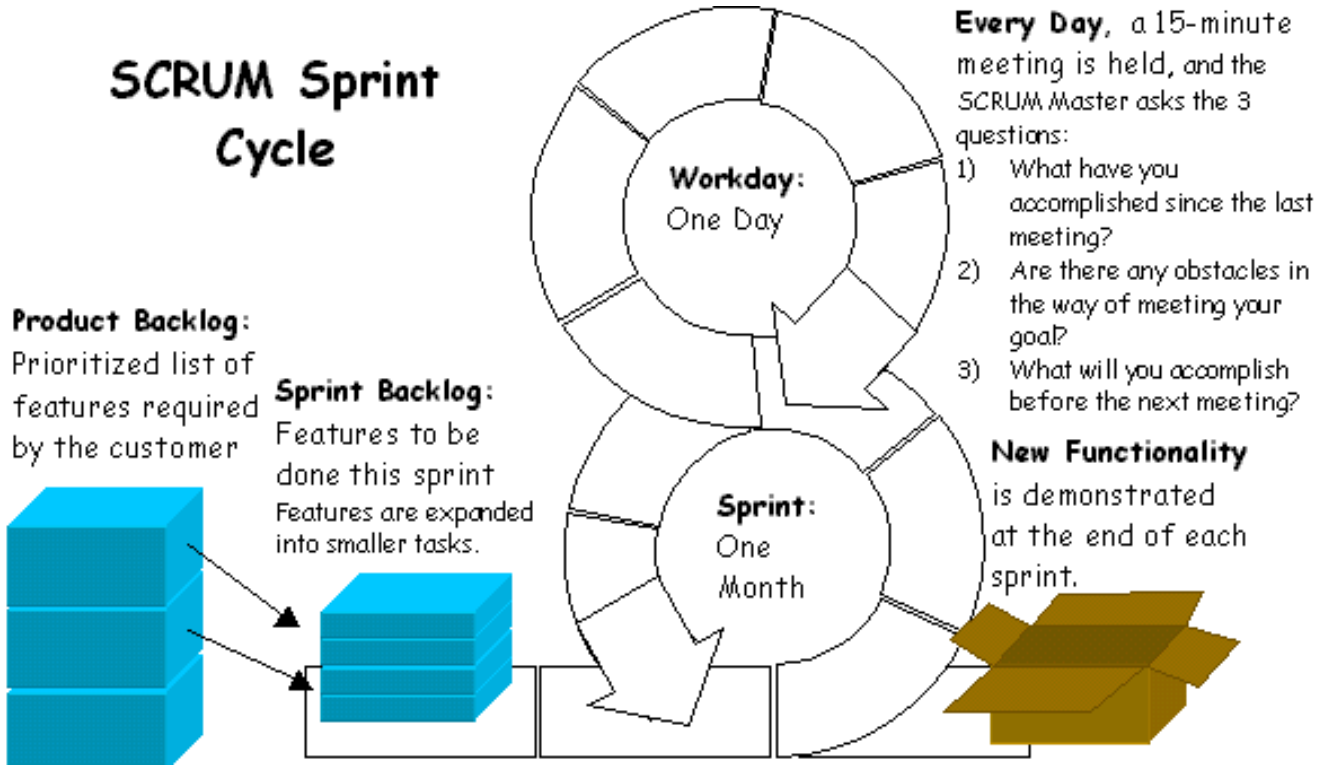
- Feedback (Small releases, Scale of minutes or days, Unit tests).
- Communication (with instructor, colleagues).
- Simplicity (What is the simplest thing that could possibly work?).
- Courage (Changing the system, Throwing code away, Pair programming).

Extreme Programming: briefly[1]



[1] http://www.extremeprogramming.org/rules/spike.html

Scrum: briefly[2]



[2] http://www.codeproject.com/Articles/4798/What-is-SCRUM

- What is academic programming?
- What tools and languages to use?
- Development models
- **Design patterns and pitfalls**
- Task

Design Patterns

For any SW developer it is highly recommended to know the design patterns. Gang of Four (GoF)[3] book is the most valued in this sense.
Patterns introduced:

1. Creational
   - Abstract Factory Pattern
   - Builder Pattern
   - Factory Method Pattern
   - Prototype Pattern
   - Singleton Pattern

[3] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", 1994.

Design Patterns

Patterns introduced:

2. Structural
   - Adapter Pattern
   - Bridge Pattern
   - Composite Pattern
   - Decorator Pattern
   - Facade Pattern
   - Flyweight Pattern
   - Proxy Pattern
3. Behavioral
   - Chain Of Responsibility Pattern
   - Command Pattern
   - Interpreter Pattern
   - Iterator Pattern
   - Mediator Pattern
   - Memento Pattern
   - Observer Pattern
   - State Pattern
   - Strategy Pattern
   - Template Method Pattern
   - Visitor Pattern

Misuse of patterns

Do not use patterns whenever you can use them

- Patterns adds complexity whenever they are not needed.
- Remember simplicity in XP.

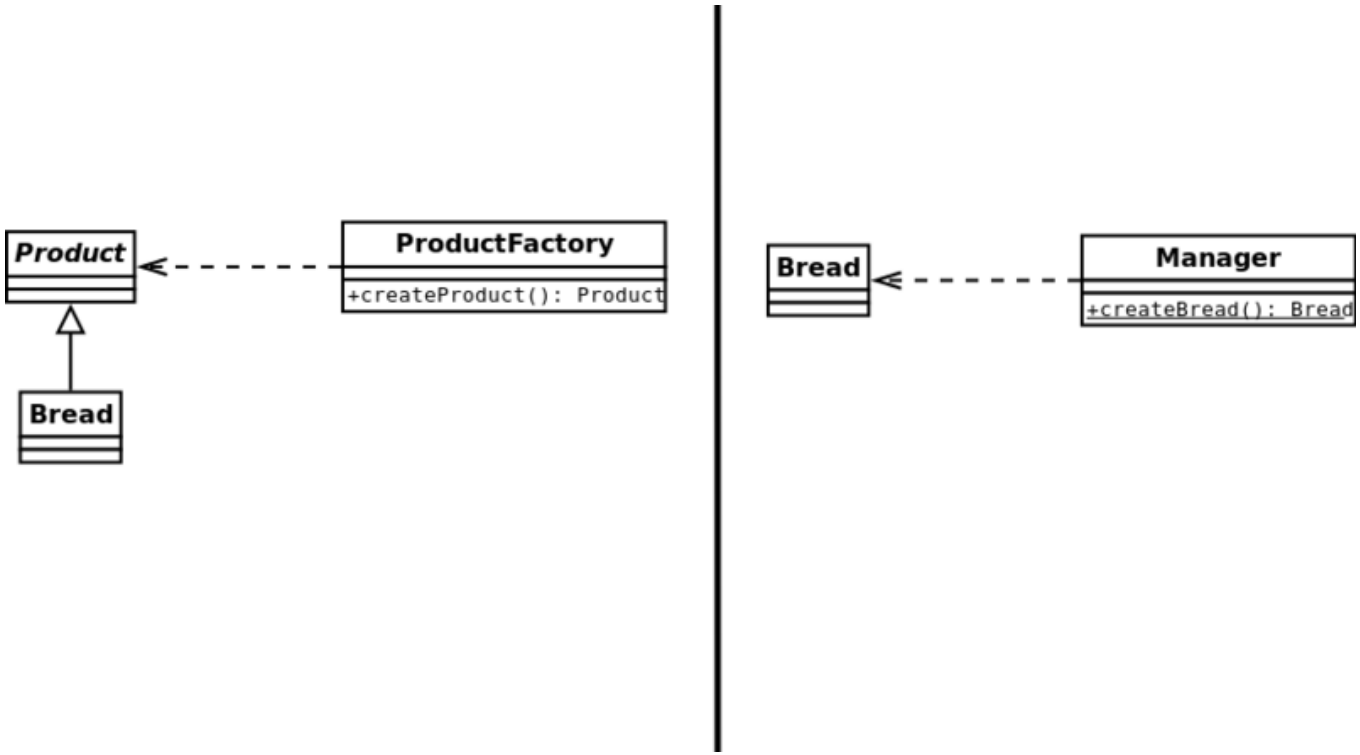
Here is example how patterns may be misused:

> Consider two workers (Alice and Bob) produce a code which Manager asks to do. Alice use a lot of patterns, Bob does not know anything about patterns. Mangers asks and changes the technical task in iteration, one by one (It is normal for Agile SW development).

Misuse of patterns

Task:

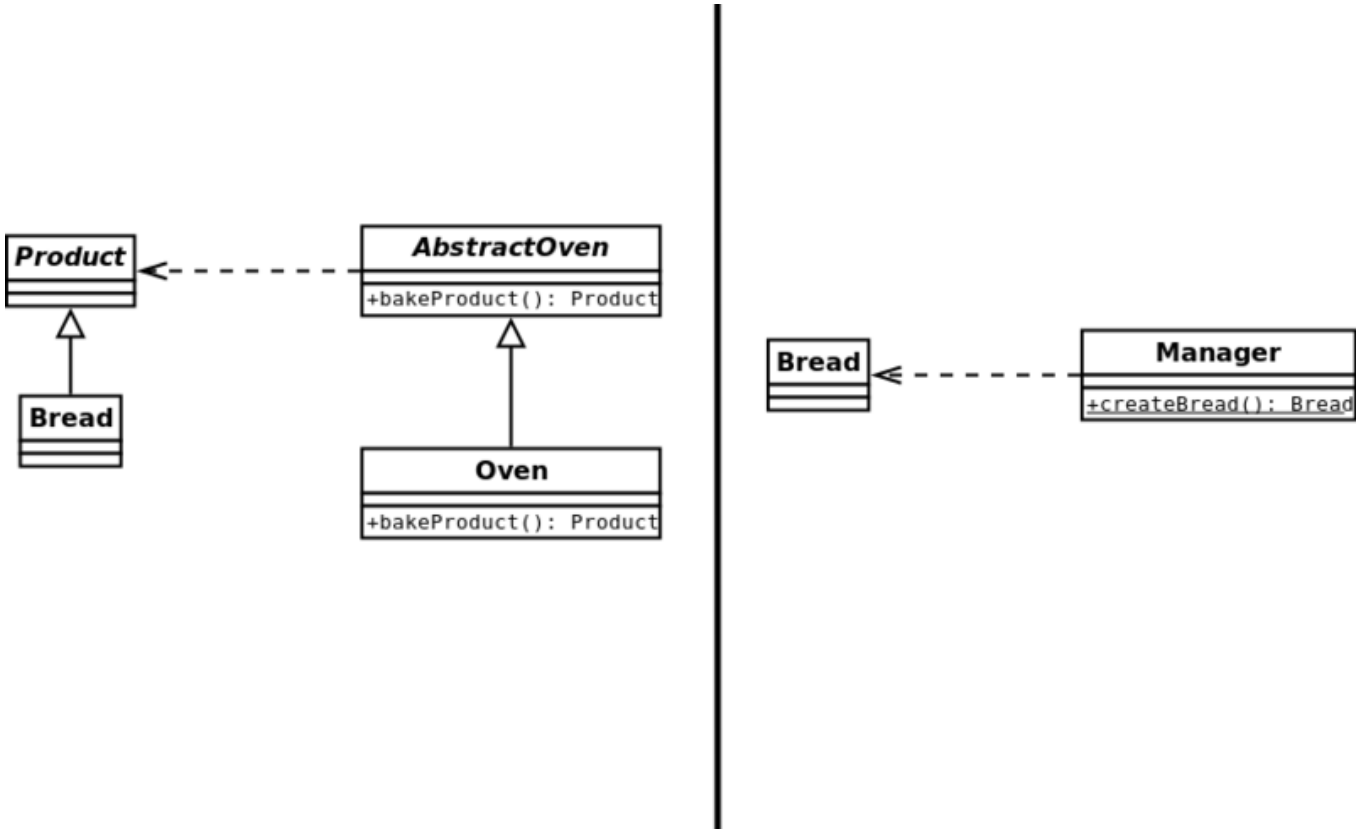
We need to produce bread.



Misuse of patterns

Task:

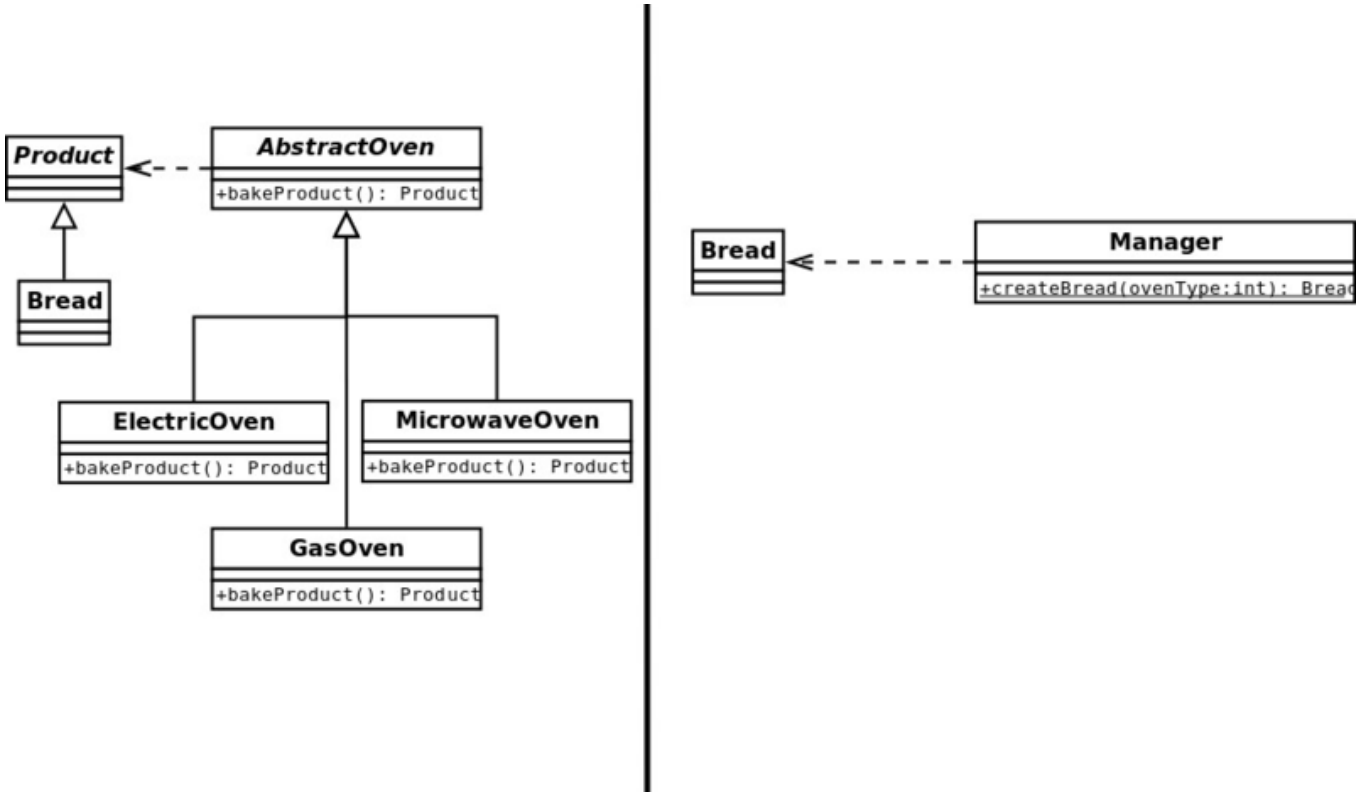
We need not just produce bread but bake it.



Misuse of patterns

Task:

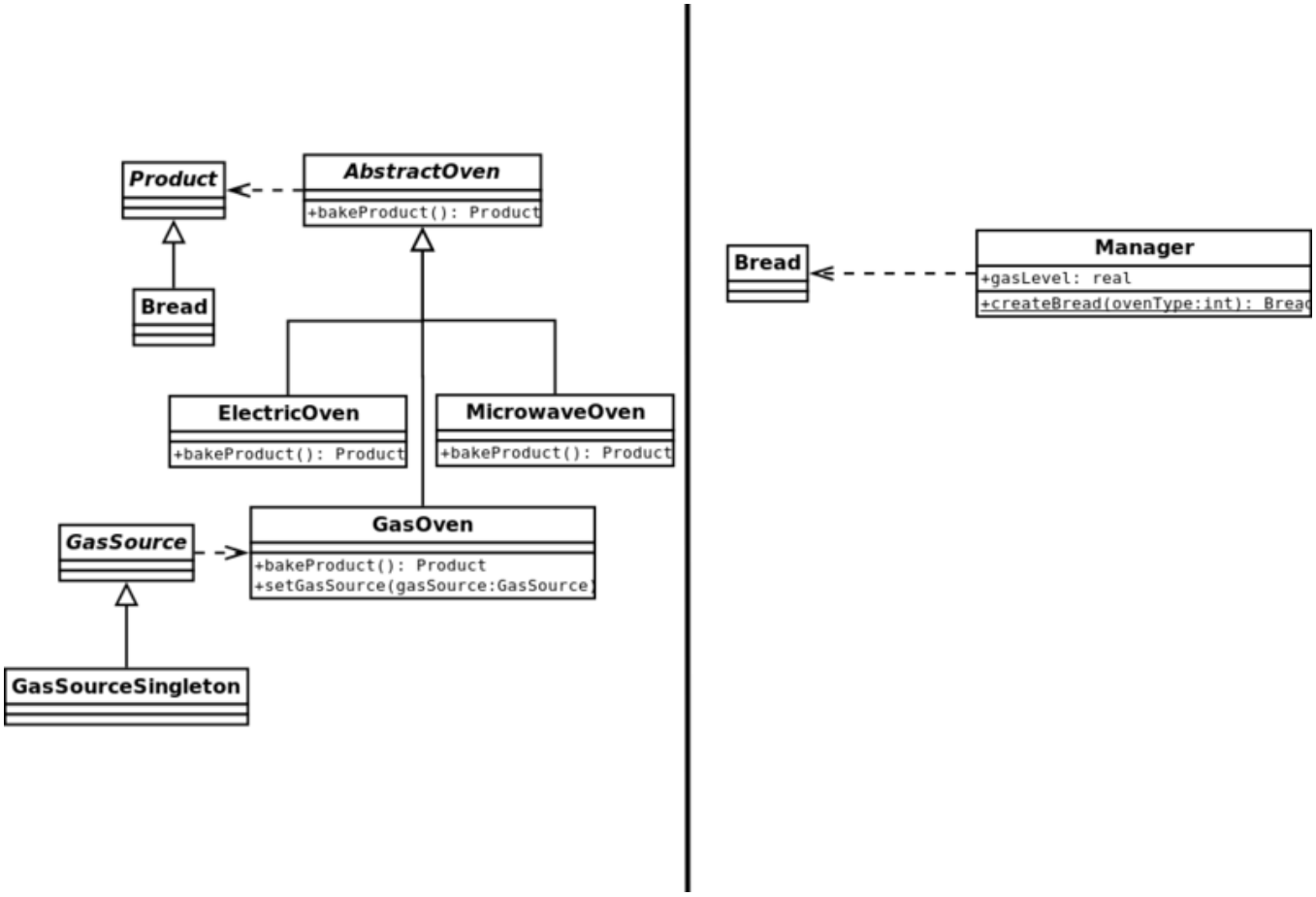
We need ovens of different types.



Misuse of patterns

Task:

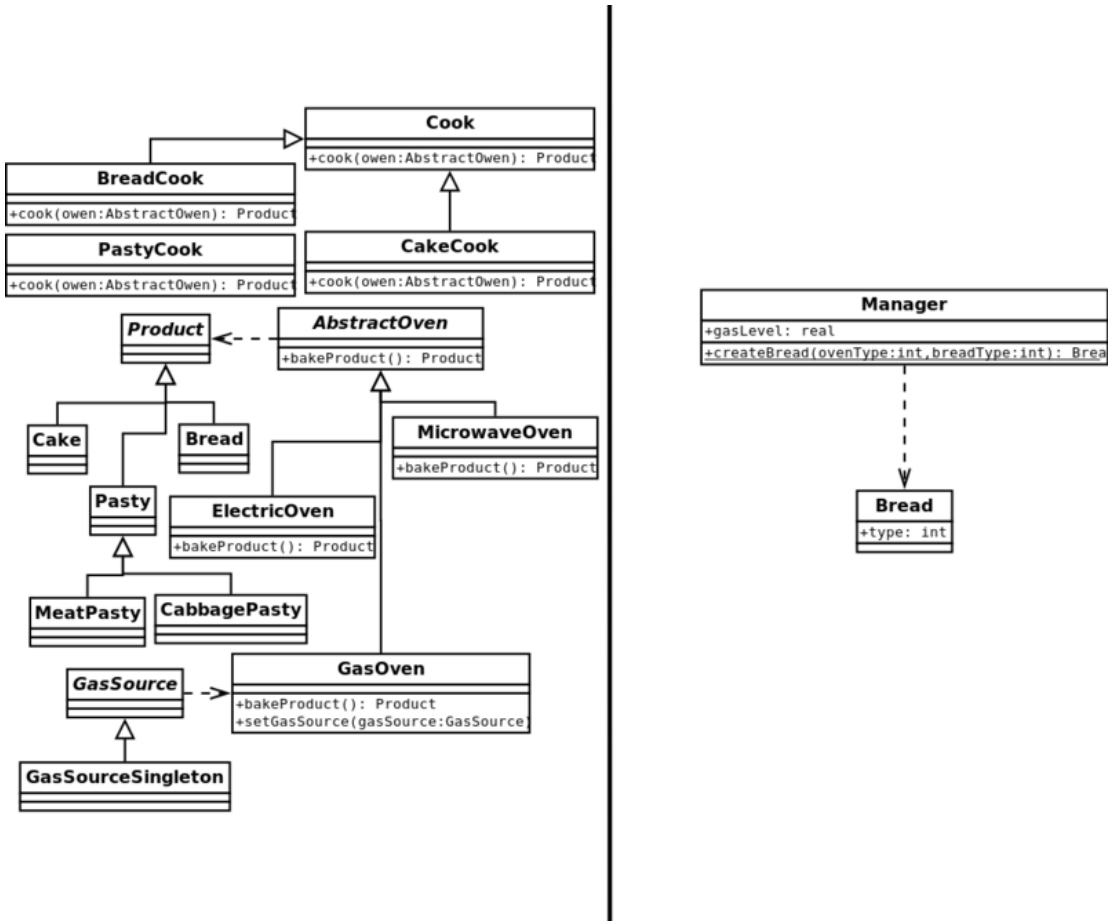
We need gas oven not to be able to bake without gas.



Misuse of patterns

Task:

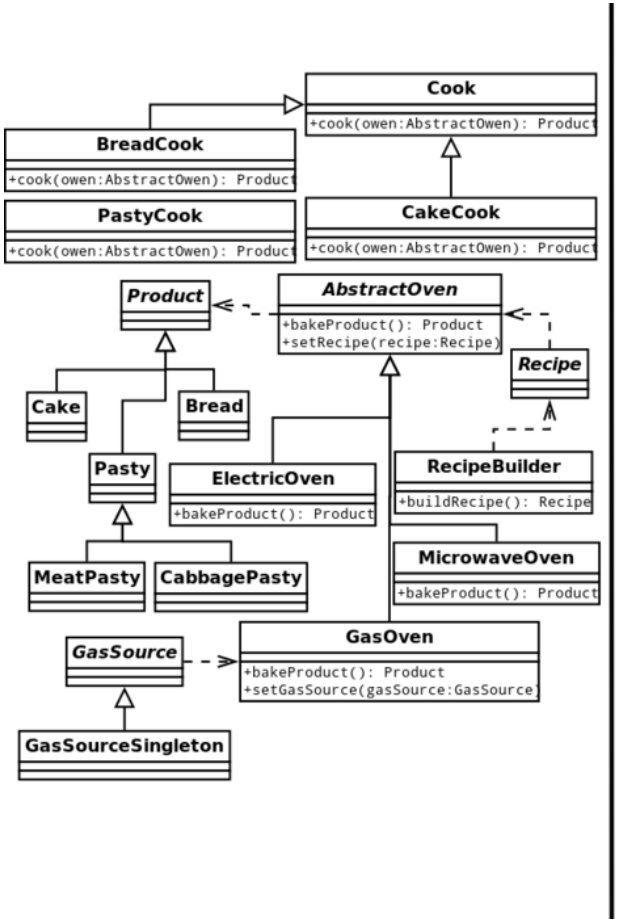
We need ovens to be able to bake cakes and pastries (with meat or with Cabbage).

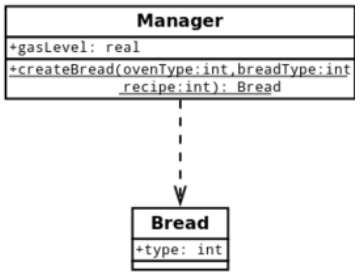


Misuse of patterns

Task:

We need add different recipes for bread, cake and pastry cooking.

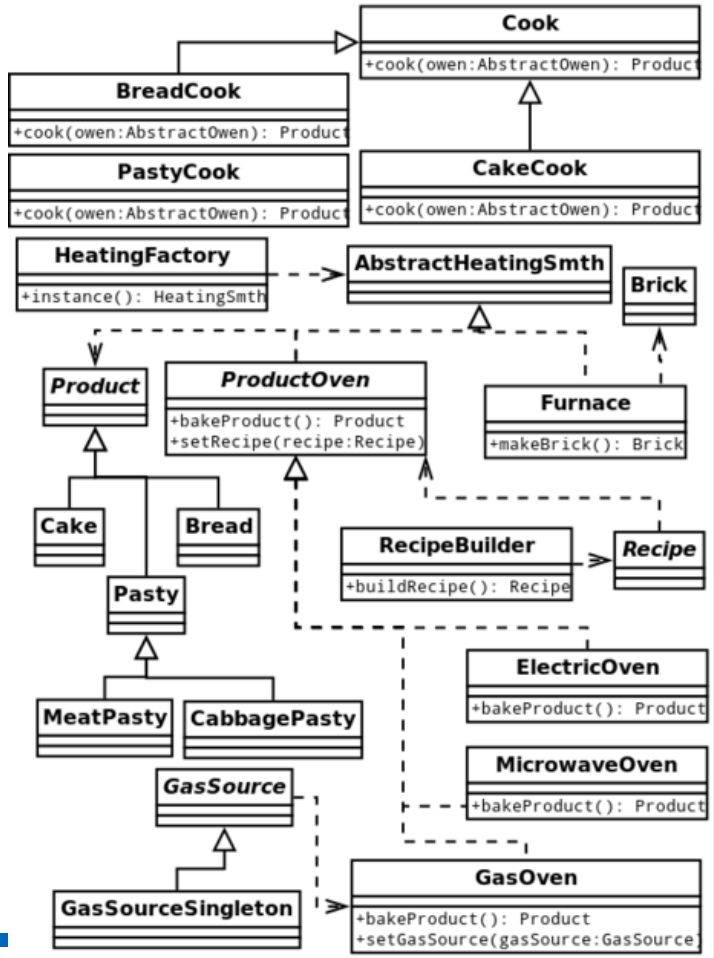


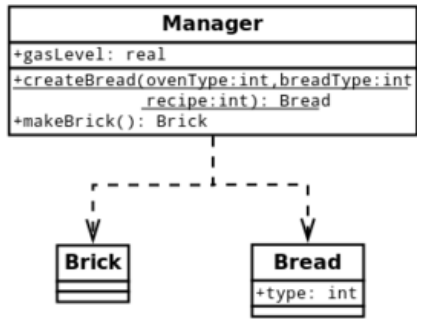


Misuse of patterns

Task:

We need the oven to be able to fire bricks.





Conclusion

Best practices for Academic programming.

- Schedule your coding/thesis writing time.
- Briefly comment and document all steps you do with the code.
- Report to instructor on a weekly basis.
- Use same coding style, if you need to reread the code you will know what it is about.
- Simplicity. Do not produce to complex code, you will forget what it does in a month.
- Use design patterns and do not use them too much.

- What is academic programming?
- What tools and languages to use?
- Development models
- Design patterns and pitfalls
- Task

Diary on Academic programming task.

Normally you do not see a lot of information on the programming process in articles. Thus, for the next diary we would like to do the following task, instead of literature report.

- Choose a tool, IDE and so on.
- With the tool evaluate your topic in any possible direction.
- Make a small program (really anything) and run the test.
- Report what you have done in the Diary.
- Report what tool/tools you have used.

Questions and Comments?

Thank you.