AspectJ

Lu Li
lu.li@liu.se

(a number of slides by)
Mikhail Chalabine

Agenda
- Problem with OOP
- AspectJ introduction
- Hints on lab3

OOP programming

- Encapsulate data and method for the same concern.

```
class BankAccount {
  int balance;
  void setBalance(int newBalance)
    { balance=newBalance; }
  int getBalance()
    { return balance; }
}
```

Problems with OOP

- Crosscutting concerns
- Logging functional requirements

```
class BankAccount {
  int balance;
  void setBalance(int newBalance)
    { balance=newBalance;
      printf("Balance changed!"); }
  int getBalance()
    { return balance;
      printf("Balance queried!"); }
}
```

Crosscutting

- Code in objects that does not relate to the functionality defined for those objects.
- Imagine adding:
  - User authentication
  - Persistence
  - Timing
  - ...

Mixing Concerns

- Mixing of concerns lead to:
  - Code scattering
  - Code tangling
Mixing Concerns

• Correctness
  ◦ Understandability
  ◦ Testability

• Maintenance
  ◦ Find code
  ◦ Change it consistently
  ◦ No help from OO tools

• Reuse

Aspect programming

• Encapsulate crosscutting concerns as a aspect

public BankAccount
{
  int balance;
  void setBalance(int newBalance)
  {
    balance = newBalance;
    printf("Balance changed!");
  }
  int getBalance()
  {
    return balance;
    printf("Balance queried!");
  }
}

Aspect programming

public aspect Log
{
  after() returning: setBalance(int newBalance)
  {
    printf("Balance changed!");
  }
  after() returning: getBalance()
  {
    printf("Balance queried!");
  }
}

Aspect programming improved

public aspect Log
{
  pointcut LogSet():call(public void setBalance(int newBalance))
  pointcut LogGet():call(public int getBalance())
  after() returning: LogSet()
  {
    printf("Balance changed!");
  }
  after() returning: LogGet()
  {
    printf("Balance queried!");
  }
}

This is aspect!!!

logging is not modularized

• logging in org.apache format
  ◦ red shows lines of code that handle logging
  ◦ not in just one place
  ◦ not even in a small number of places

Aspect programming

public aspect Log
{
  pointcut logSet():call(public void setBalance(int newBalance))
  pointcut logGet():call(public int getBalance())
  after() returning: LogSet()
  {
    printf("Balance changed!");
  }
  after() returning: LogGet()
  {
    printf("Balance queried!");
  }
}

Weaving

Components in component language

Weaver

Executable program

Aspects in aspect language

Join point

Advice

This is aspect!!!
**Weaving time**
- Preprocessor
- Compile time
- Link time
- Load time
- Run time

**Weaving in BankAccount**

**Formal definition: Join Point**
- A location in (component) code where a concern crosscuts (static join point model)
  - Method declaration
    - public void Account.deposit(int)
  - A well-defined point in the program flow (dynamic join point model, e.g., in AspectJ)
    - A call to a method etc.
    - call(public void Account.deposit(int))

**Pointcut**
- A pointcut picks out certain join points
- In the following the balanceAltered pointcut picks out each join point that is a call to either the deposit() or the withdraw() method of an Account class

```java
pointcut balanceAltered() : call(public Account.deposit(int)) || call(public Account.withdraw(int));
```

**Pointcut (further examples)**
- call(void SomeClass.make*(..))
  - picks out each join point that's a call to a void method defined on SomeClass whose name begins with "make" regardless of the method's parameters
- call(public SomeClass.* (..))
  - picks out each call to SomeClass's public methods
- cflow(somePointcut)
  - picks out each pointcut that occurs in the dynamic context of the join points picked out by somePointcut pointcuts in the control flow, e.g., in a chain of method calls

**cflow**

```java
f1()
cflow(call void f1())
foo()
f2()
```
A piece of Advice

- Code that is executed at a pointcut (when a join point is reached)

```java
before(int i) : balanceAltered(i) {
    System.out.println("The balance changed");
}
```

Advice

- Before
- After
- Around

Around advice

```java
void around(): safeCheck() {
    ...
    proceed();
    ...
}
```

Aspect

- Groups join points, pointcuts and advice.
- The unit of modularity for a crosscutting concern.

```java
public aspect LoggingAspect {
    pointcut balanceAltered() :
    call(public void Account.deposit(int)) ||
    call(public void Account.withdraw(int));

    before(int i) : balanceAltered(i) {
        System.out.println("The balance changed");
    }
}
```

Exposing context to advice

- AspectJ facility

```java
pointcut raise(Employee emp, int amount) : call(void raiseSalary(int)) &&
    target(emp) &&
    args(amount);

before(Employee emp, int amount) : raise(emp, amount) {
    System.out.println("to receive amount of " + amount);
}
```

Exposing context to advice

- The thisJoinPoint model

```java
getThis()
getTarget()
getArgs()
getSignature()
getSourceLocation()
getKind()
toString()
toShortString()
toLongString()
```
The thisJoinPoint model, an example

```java
public class DVD extends Product {
    private String title;
    ...
}

sourceLocation sl = thisJoinPoint.getSourceLocation();
Class theClass = (Class) sl.getWithInType();
System.out.println(theClass.toString());

Output: class DVD
```

Weaving in class inheritance hierarchy

- If not specified (call(public void f())), weave for all. (all have the same method name)
- If specified (call(public void SomeClass.f()))
  - SomeClass is the father, all children will be weaved even though the child override the f() method.
  - SomeClass is the child, only the child is weaved.

Other tools

- AspectWerkz
- JAC
- JBoss-AOP
- AspectJ
- LOOM.NET
- AspectR
- AspectS
- AspectC
- AspectC++
- Pythius
- J2EE Interceptors

Tips for lab3

- An naïve version of class BankAccount and it should be kept unchanged
- By aspectJ, mix crosscutting concerns “Safety check” with BankAccount.withdraw(float)
- After weaving, safety check will be performed at the right timing of calling withdraw(float)

Deadlines for 3 labs

- Friday, May 24 2013
- Important if you want 4 bonus points in the exam.

Questions?