Politechnika Wrocławska

Wzorce Projektowe

dr inż. Paweł Trajdos

Politechnika Wrocławska, Katedra Systemów i Sieci Komputerowych
Wyb. Wyspianskiego 27, 50-370 Wroclaw

5 lutego 2023

Politechnika Wrocławska

Spis treści

Subsection 1

Visitor

Idea

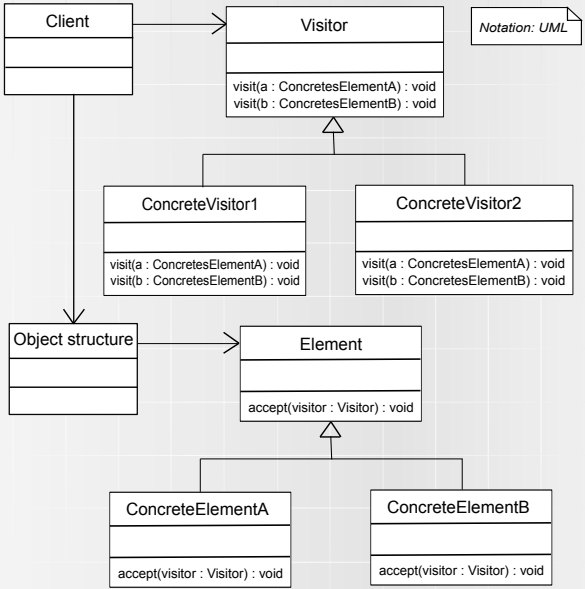

Client
Visitor
Notation: UML
visit(a : ConcretesElementA) : void
visit(b : ConcretesElementB) : void
ConcreteVisitor1
visit(a : ConcretesElementA) : void
visit(b : ConcretesElementB) : void
ConcreteVisitor2
visit(a : ConcretesElementA) : void
visit(b : ConcretesElementB) : void
Object structure
Element
accept(visitor : Visitor) : void
ConcreteElementA
accept(visitor : Visitor) : void
ConcreteElementB
accept(visitor : Visitor) : void

Diagram klas

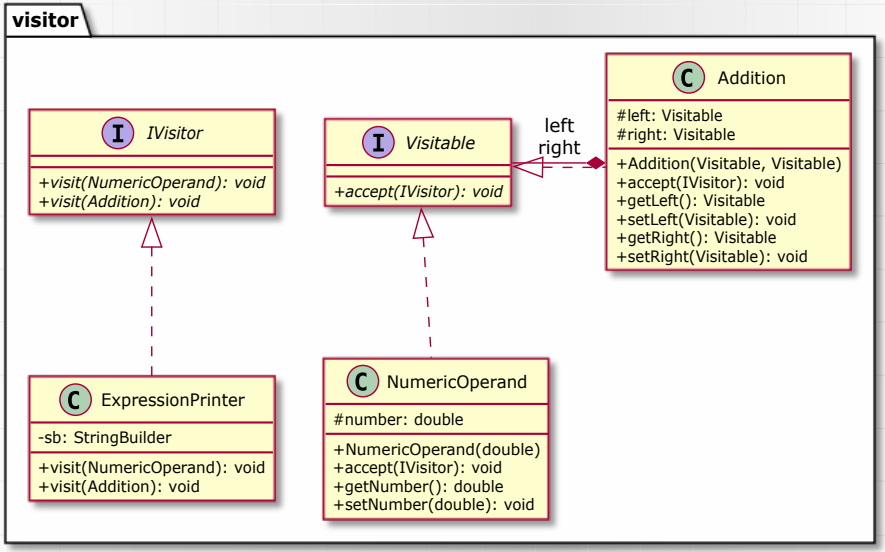


Diagram obiektów

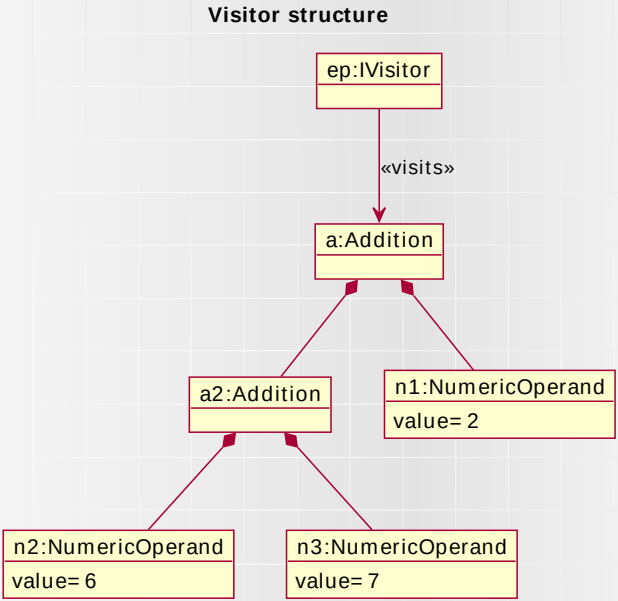
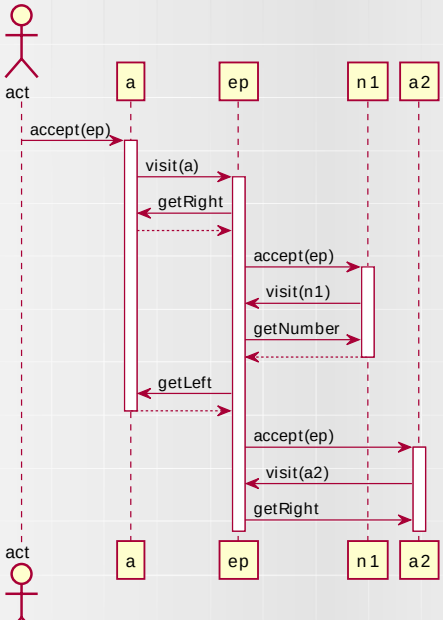


Diagram sekwencji



Implementacja

Listing: Visitable.java

```
1 package visitor;
2
3 public interface Visitable {
4   public void accept(IVisitor visitor);
5 }
```

Implementacja

Listing: IVisitor.java

```
1  package visitor;
2
3  public interface IVisitor {
4    public void visit(NumericOperand oper);
5    public void visit(Addition oper);
6
7  }
```

Implementacja

Listing: NumericOperand.java

```java
package visitor;

public class NumericOperand implements Visitable {
  protected double number;
  public NumericOperand(double number) {this.number = number; }
  @Override
  public void accept(IVisitor visitor) {
    visitor.visit(this);
  }
  public double getNumber() {
    return number;
  }
  public void setNumber(double number) {
    this.number = number;
  }
}
```

Implementacja

Listing: Addition.java

```
1  package visitor;
2
3  public class Addition implements Visitable {
4
5    protected Visitable left;
6    protected Visitable right;
7
8    public Addition(Visitable left, Visitable right) {this.left = left;this.right=right;}
9    @Override
10   public void accept(IVisitor visitor) {visitor.visit(this);}
11   public Visitable getLeft() {
12     return left;
13   }
14   public void setLeft(Visitable left) {
15     this.left = left;
16   }
17   public Visitable getRight() {
18     return right;
19   }
20   public void setRight(Visitable right) {
21     this.right = right;
22   }
```

Implementacja

Listing: ExpressionPrinter.java

```
1  package visitor;
2
3  public class ExpressionPrinter implements IVisitor {
4
5    private StringBuilder sb;
6    public ExpressionPrinter() {
7      sb = new StringBuilder();
8    }
9    @Override
10   public void visit(NumericOperand oper) {sb.append(""+oper.number);}
11   @Override
12   public void visit(Addition oper) {
13     sb.append("(");
14     oper.getLeft().accept(this);
15     sb.append("+");
16     oper.getRight().accept(this);
17     sb.append(")");
18   }
19   @Override
20   public String toString() {return sb.toString();}
21 }
```

Politechnika Wrocławska

Subsection 2

Chain of responsibilities

Idea

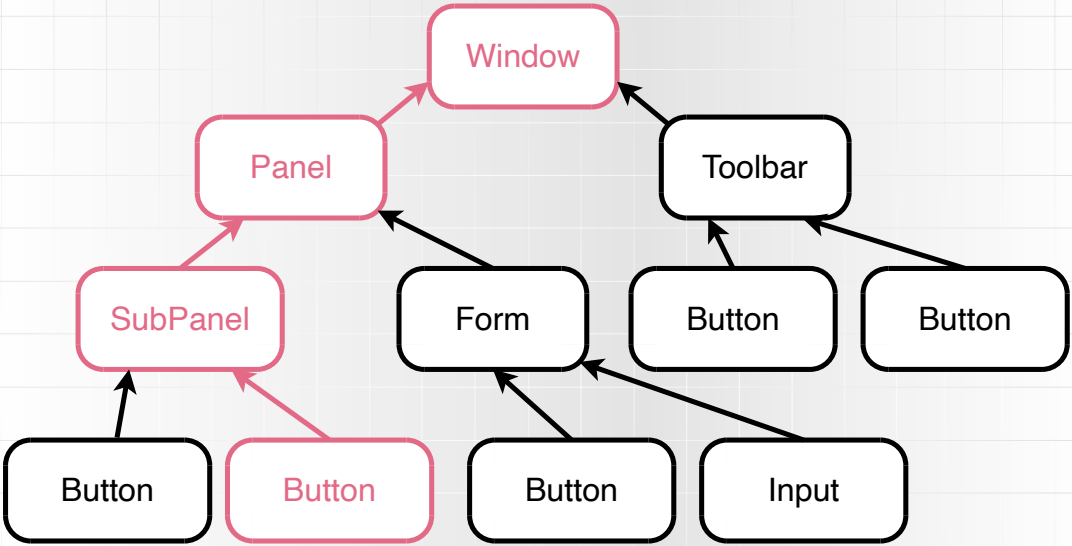
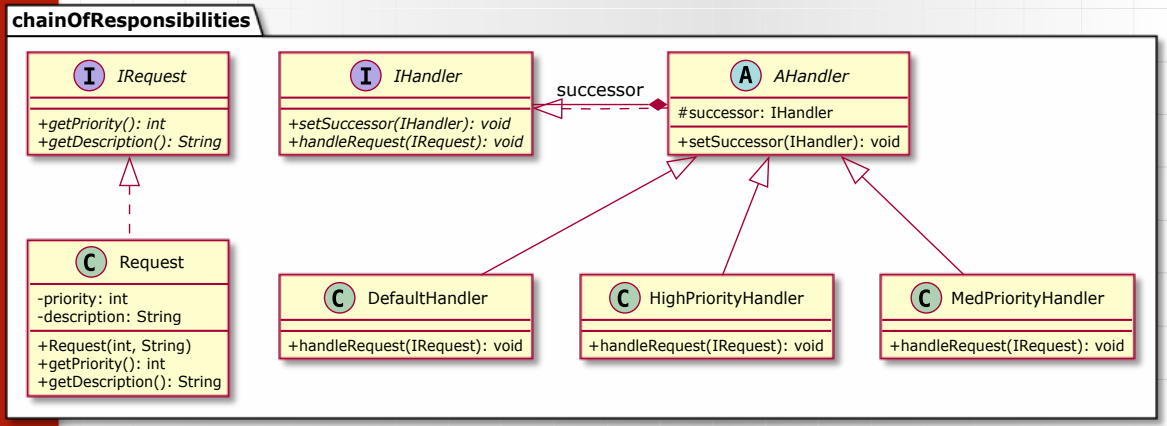

Window
Panel
Toolbar
SubPanel
Form
Button
Button
Button
Button
Button
Input

Diagram klas



Politechnika Wrocławska

Diagram obiektów

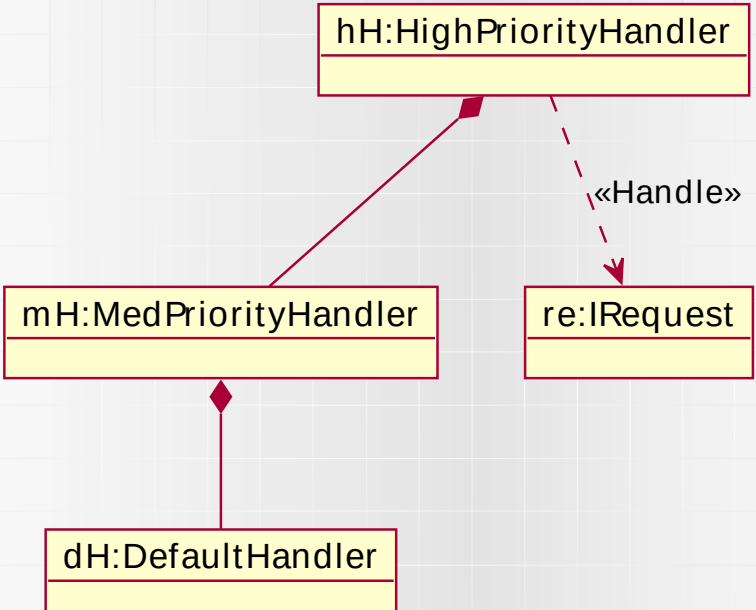
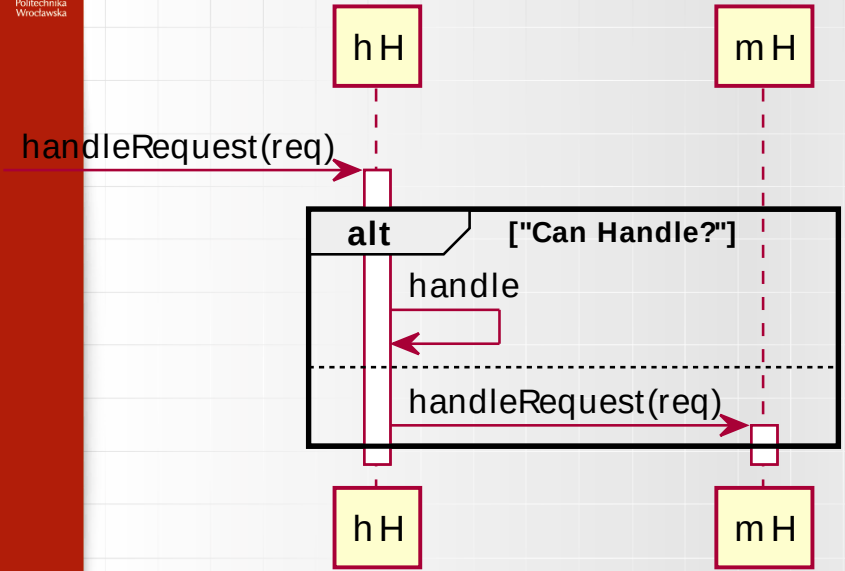


Diagram sekwencji



Politechnika Wrocławska

Implementacja

Listing: IRequest.java

```
1  package chainOfResponsibilities;
2
3  public interface IRequest {
4
5    public int getPriority();
6    public String getDescription();
7
8  }
```

Implementacja

Listing: Request.java

```java
package chainOfResponsibilities;

public class Request implements IRequest {

  private int priority;
  private String description;

  public Request(int priority, String description) {
    this.priority = priority;
    this.description = description;
  }
  @Override
  public int getPriority() {return priority;  }
  @Override
  public String getDescription() { return description;}
  @Override
  public String toString() {
    return "Request: " + description +" ;priority: " + priority;
  }

}
```

Implementacja

Listing: IHandler.java

```
1  package chainOfResponsibilities;
2
3  public interface IHandler {
4
5    public void setSuccessor(IHandler handler);
6    public void handleRequest(IRequest request);
7
8  }
```

Politechnika Wrocławska

Implementacja

Listing: AHandler.java

```
1  package chainOfResponsibilities;
2
3  public abstract class AHandler implements IHandler {
4
5    protected IHandler successor;
6    @Override
7    public void setSuccessor(IHandler handler) { successor = handler; }
8  }
```

Politechnika Wrocławska

Implementacja

Listing: HighPriorityHandler.java

```
1  package chainOfResponsibilities;
2
3  public class HighPriorityHandler extends AHandler {
4    @Override
5    public void handleRequest(IRequest request) {
6      if(request.getPriority()<=0) {
7        System.out.println("Handling High priority Request" + request);
8      }else {
9        if(successor!=null)successor.handleRequest(request);
10     }
11   }
12 }
```

Implementacja

Listing: MedPriorityHandler.java

```
1  package chainOfResponsibilities;
2
3  public class MedPriorityHandler extends AHandler {
4    @Override
5    public void handleRequest(IRequest request) {
6      int priority = request.getPriority();
7      if(priority>0 & priority< 100 ) {
8        System.out.println("Handling Medium priority Request" + request);
9      }else {
10       if(successor!=null)successor.handleRequest(request);
11     }
12   }
13 }
```

Politechnika Wrocławska

Implementacja

Listing: DefaultHandler.java

```java
package chainOfResponsibilities;

public class DefaultHandler extends AHandler {
  @Override
  public void handleRequest(IRequest request) {
    System.out.println("Default Handler: " + request);
  }
}
```

Tests

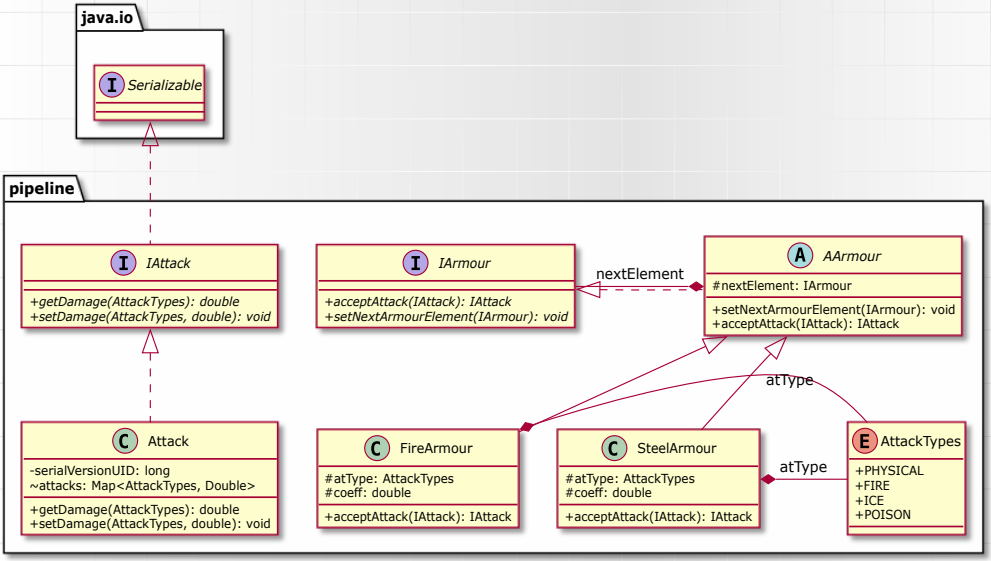
Listing: HandlersTest.java

```java
package chainOfResponsibilities;

import org.junit.Test;

public class HandlersTest {

  @Test
  public void test() {
    IRequest req = new Request(5, "A med priority request");
    IHandler highH = new HighPriorityHandler();
    IHandler medH = new MedPriorityHandler();
    IHandler defH = new DefaultHandler();

    highH.setSuccessor(medH);
    medH.setSuccessor(defH);

    highH.handleRequest(req);

  }

}
```

Subsection 3

Chain of responsibilities – wersja zmodyfikowana

Diagram klas



Politechnika Wrocławska

Diagram obiektów

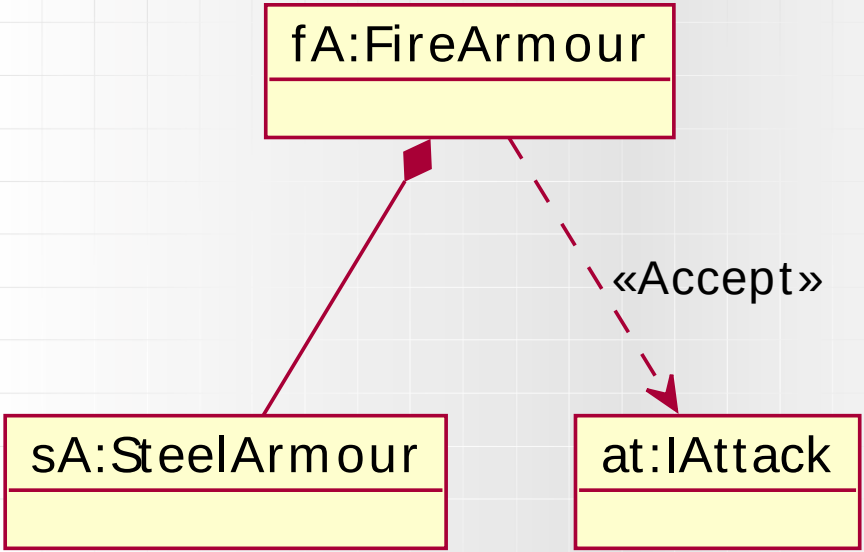
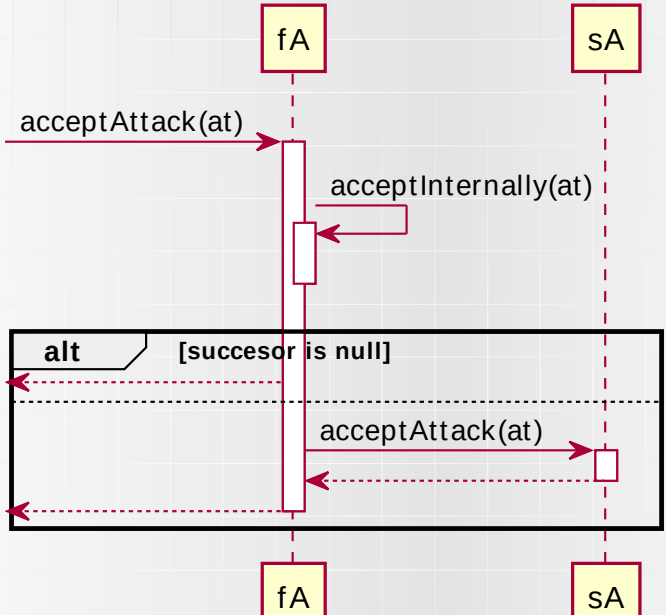


Diagram sekwencji



Politechnika Wrocławska

Implementacja

Listing: IArmour.java

```
1  package pipeline;
2
3
4  public interface IArmour {
5
6    public IAttack acceptAttack(IAttack attack);
7    public void setNextArmourElement(IArmour armour);
8
9  }
```

Implementacja

Listing: AArmour.java

```java
package pipeline;
import org.apache.commons.lang3.SerializationUtils;



public abstract class AArmour implements IArmour {
  protected IArmour nextElement;
  @Override
  public void setNextArmourElement(IArmour armour) {nextElement = armour;}
  @Override
  public IAttack acceptAttack(IAttack attack) {
    return SerializationUtils.clone(attack);
  }
}
```

Implementacja

Listing: SteelArmour.java

```
1  package pipeline;
2
3
4  public  class SteelArmour extends AArmour {
5    protected AttackTypes atType = AttackTypes.PHYSICAL;
6    protected double coeff = 0.2;
7    @Override
8    public IAttack acceptAttack(IAttack attack) {
9      IAttack tmpAttack = super.acceptAttack(attack);
10     double attackVal =tmpAttack.getDamage(atType);
11     if(attackVal>0) {
12       attackVal= (1-coeff)*attackVal;
13       tmpAttack.setDamage(atType, attackVal);
14     }
15     if(nextElement != null)tmpAttack = nextElement.acceptAttack(tmpAttack);
16     return tmpAttack;
17   }
18 }
```

Implementacja

Listing: FireArmour.java

```
1  package pipeline;
2
3
4  public  class FireArmour extends AArmour {
5    protected AttackTypes atType = AttackTypes.FIRE;
6    protected double coeff = 0.2;
7    @Override
8    public IAttack acceptAttack(IAttack attack) {
9      IAttack tmpAttack = super.acceptAttack(attack);
10     double attackVal =tmpAttack.getDamage(atType);
11     if(attackVal>0) {
12       attackVal= (1-coeff)*attackVal;
13       tmpAttack.setDamage(atType, attackVal);
14     }
15     if(nextElement != null)tmpAttack = nextElement.acceptAttack(tmpAttack);
16     return tmpAttack;
17   }
18 }
```

Tests

Listing: SteelArmourTest.java

```java
package pipeline;
import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class SteelArmourTest {
  @Test
  public void test() {
    IAttack att = new Attack();
    att.setDamage(AttackTypes.PHYSICAL, 10.0);
    att.setDamage(AttackTypes.FIRE, 100.0);
    att.setDamage(AttackTypes.POISON, 10.0);
    IArmour sA = new SteelArmour();
    IArmour fA = new FireArmour();
    fA.setNextArmourElement(sA);
    IAttack handledAttack = fA.acceptAttack(att);
    assertEquals("Physical", 8.0, handledAttack.getDamage(AttackTypes.PHYSICAL),1E-5);
    assertEquals("Fire", 80.0, handledAttack.getDamage(AttackTypes.FIRE),1E-5);
    assertEquals("Poison", 10, handledAttack.getDamage(AttackTypes.POISON),1E-5);
  }
}
```

Politechnika Wrocławska

Subsection 4

Template Method

Diagram klas

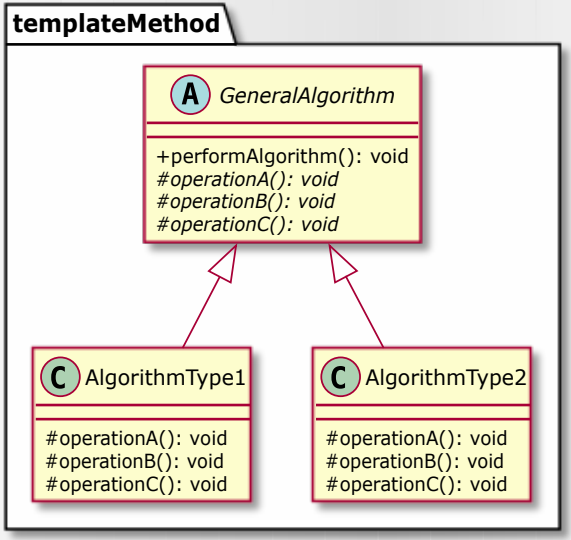
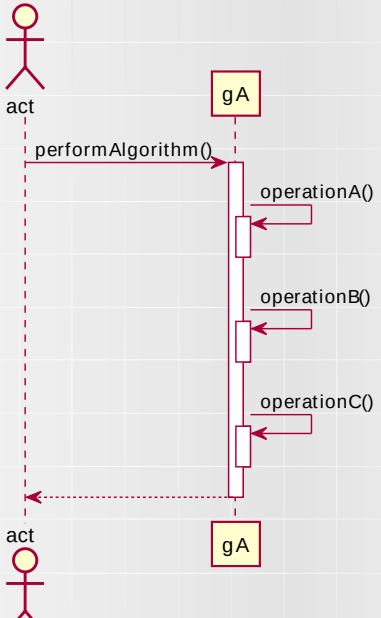


Diagram sekwencji



Implementacja

Listing: GeneralAlgorithm.java

```
1  package templateMethod;
2
3  public abstract class GeneralAlgorithm {
4
5    public final void performAlgorithm() {
6      operationA();
7      operationB();
8      operationC();
9    }
10
11   protected abstract void operationA();
12   protected abstract void operationB();
13   protected abstract void operationC();
14
15 }
```

Implementacja

Listing: AlgorithmType1.java

```java
package templateMethod;

public class AlgorithmType1 extends GeneralAlgorithm {
  @Override
  protected void operationA() {
    // TODO Auto-generated method stub
  }
  @Override
  protected void operationB() {
    // TODO Auto-generated method stub
  }
  @Override
  protected void operationC() {
    // TODO Auto-generated method stub
  }
}
```

Politechnika Wrocławska

Subsection 5

Strategy

Diagram klas

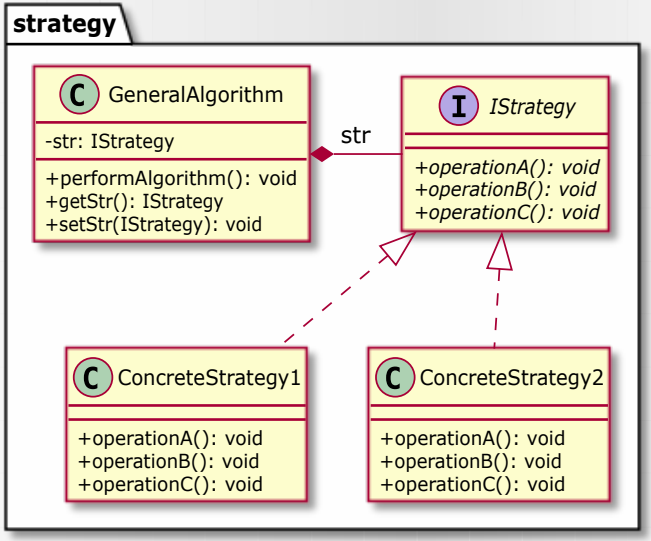
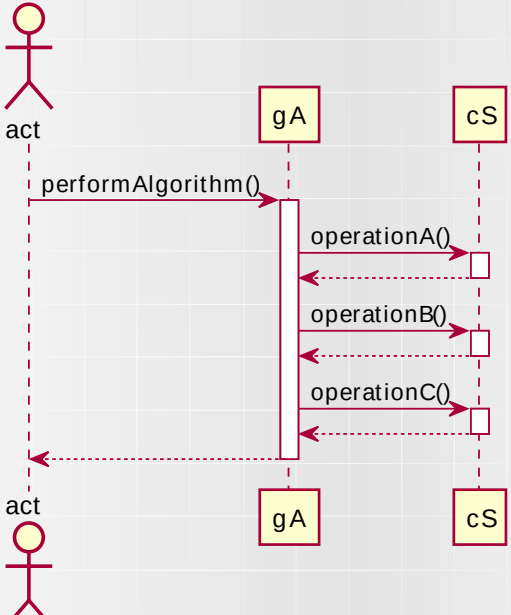


Diagram sekwencji



Implementacja

Listing: GeneralAlgorithm.java

```
1  package strategy;
2
3  public class GeneralAlgorithm {
4
5    private IStrategy str;
6
7    public final void performAlgorithm() {
8      str.operationA();
9      str.operationB();
10     str.operationC();
11   }
12   public IStrategy getStr() { return str; }
13
14   public void setStr(IStrategy str) { this.str = str; }
15
16
17 }
```

Implementacja

Listing: IStrategy.java

```
1 package strategy;
2
3 public interface IStrategy {
4   public void operationA();
5   public void operationB();
6   public void operationC();
7 }
```

Implementacja

Listing: ConcreteStrategy1.java

```java
package strategy;

public class ConcreteStrategy1 implements IStrategy {
  @Override
  public void operationA() {
    // TODO Auto-generated method stub
  }
  @Override
  public void operationB() {
    // TODO Auto-generated method stub
  }
  @Override
  public void operationC() {
    // TODO Auto-generated method stub
  }
}
```

Politechnika Wrocławska

Subsection 6

State

Idea

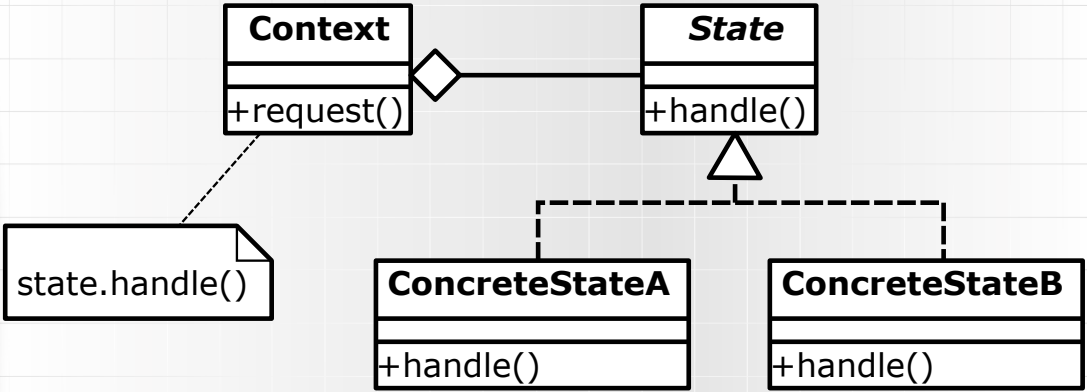

Context
+request()
State
+handle()
state.handle()
ConcreteStateA
+handle()
ConcreteStateB
+handle()

Diagram klas

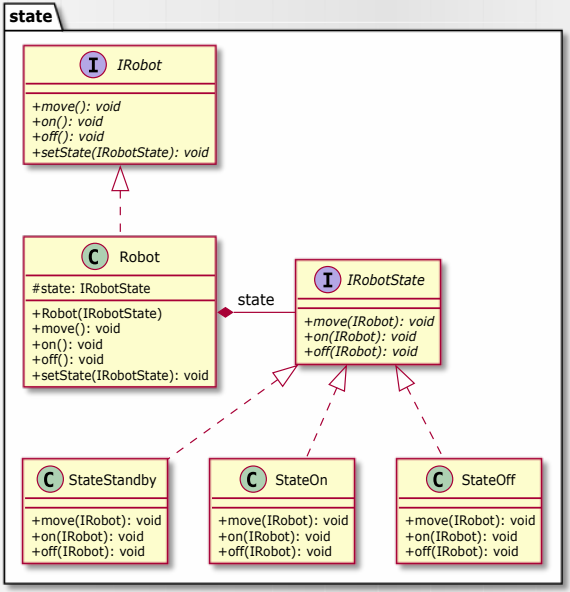
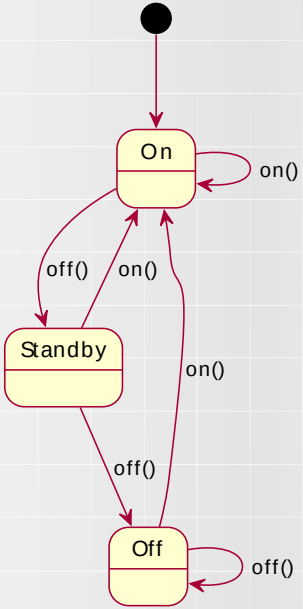


Diagram stanów



Implementacja

Listing: IRobot.java

```
1  package state;
2
3  public interface IRobot {
4
5    public void move();
6    public void on();
7    public void off();
8    public void setState(IRobotState state);
9
10 }
```

Implementacja

Listing: IRobotState.java

```
1  package state;
2
3  public interface IRobotState {
4
5    public void move(IRobot robot);
6    public void on(IRobot robot);
7    public void off(IRobot robot);
8
9  }
```

Implementacja

Listing: Robot.java

```
1  package state;
2
3  public class Robot implements IRobot {
4
5    protected IRobotState state;
6
7    public Robot(IRobotState initState) {state=initState; }
8    @Override
9    public void move() {state.move(this);}
10   @Override
11   public void on() {state.on(this);}
12   @Override
13   public void off() {state.off(this);}
14   @Override
15   public void setState(IRobotState state) {this.state = state; }
16 }
```

Implementacja

Listing: StateOn.java

```
1  package state;
2
3  public class StateOn implements IRobotState {
4    @Override
5    public void move(IRobot robot) {System.out.println("Moving....");}
6    @Override
7    public void on(IRobot robot) {robot.setState(this);}
8    @Override
9    public void off(IRobot robot) {robot.setState(new StateStandby());}
10
11 }
```

Politechnika Wrocławska

Implementacja

Listing: StateOff.java

```
1  package state;
2
3  public class StateOff implements IRobotState {
4    @Override
5    public void move(IRobot robot) {}
6    @Override
7    public void on(IRobot robot) {robot.setState(new StateOn());}
8    @Override
9    public void off(IRobot robot) {robot.setState(this);}
10
11 }
```

Implementacja

Listing: StateStandby.java

```
1  package state;
2
3  public class StateStandby implements IRobotState{
4    @Override
5    public void move(IRobot robot) {System.out.println("Standby state. Can't move!");}
6    @Override
7    public void on(IRobot robot) {robot.setState(new StateOn());}
8    @Override
9    public void off(IRobot robot) {robot.setState(new StateOff());}
10
11 }
```

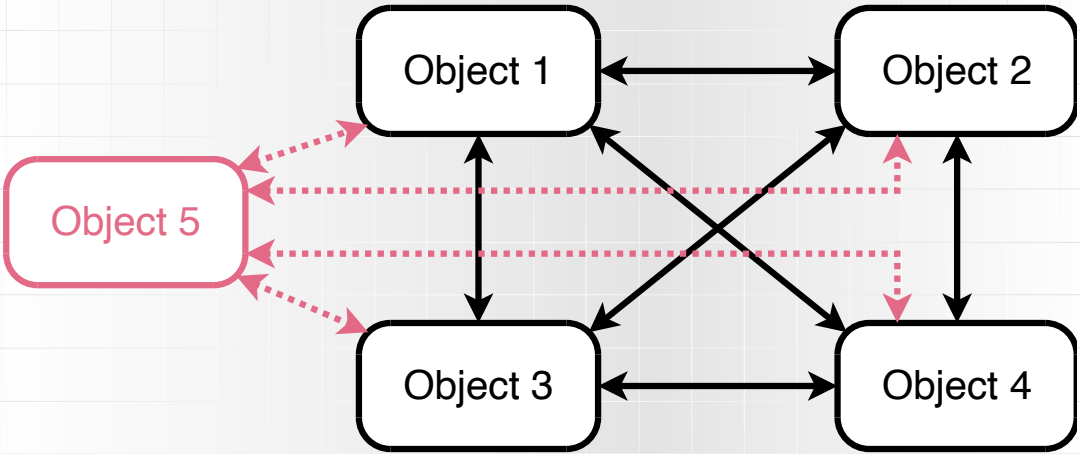
Politechnika Wrocławska

Subsection 7

Mediator

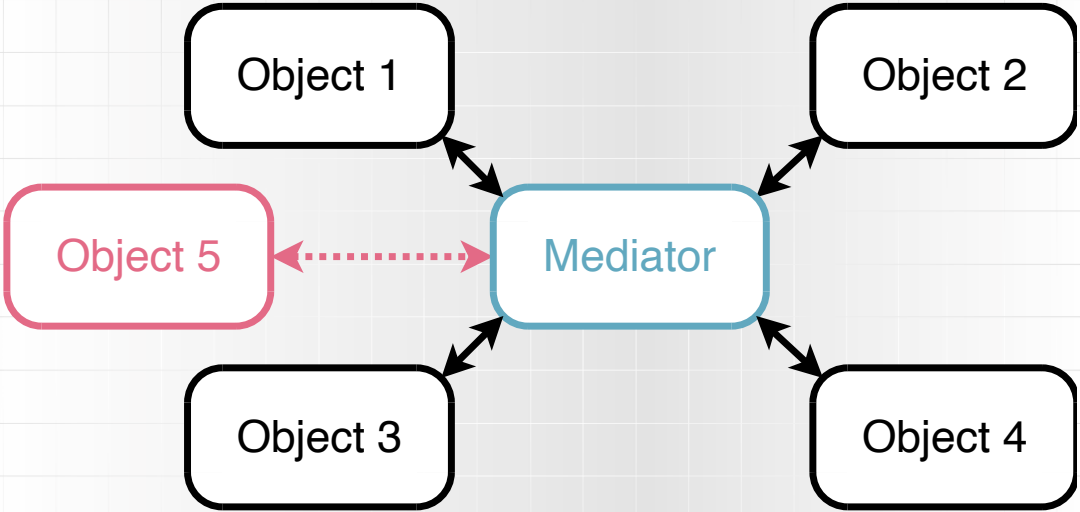
Politechnika Wrocławska

Idea

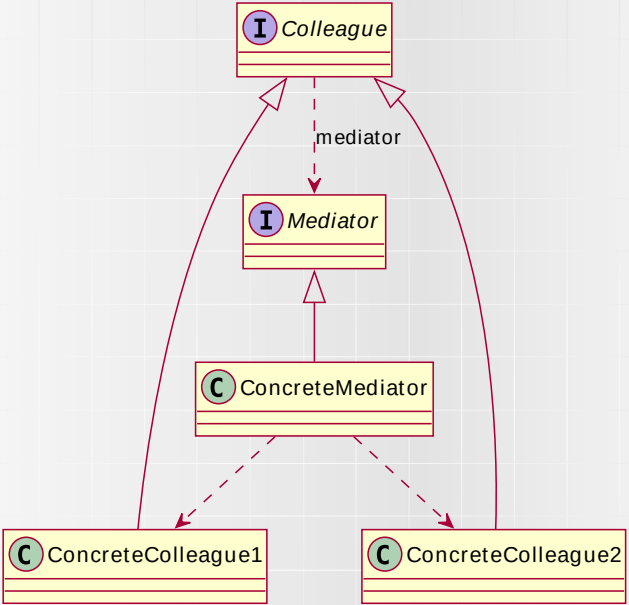


Politechnika Wrocławska

Idea

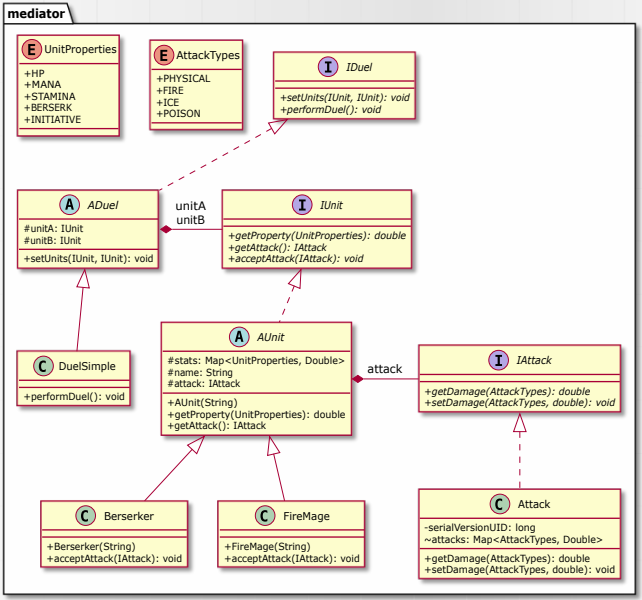


Idea


Colleague
mediator
Mediator
ConcreteMediator
ConcreteColleague1
ConcreteColleague2

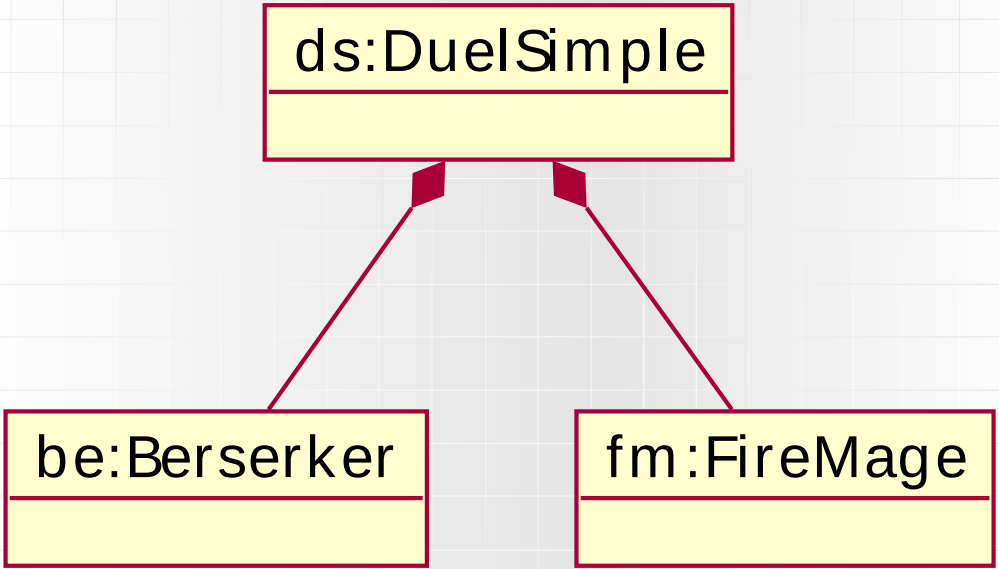
Politechnika Wrocławska

Diagram klas



Politechnika Wrocławska

Diagram obiektów



Implementacja

Listing: IAttack.java

```
1  package mediator;
2
3  public interface IAttack {
4
5    public double getDamage(AttackTypes type);
6    public void setDamage(AttackTypes type, double value);
7
8  }
```

Implementacja

Listing: Attack.java

```
1  package mediator;
2
3  import java.util.HashMap;
4  import java.util.Map;
5
6  public class Attack implements IAttack {
7
8    private static final long serialVersionUID = -1692503483936934114L;
9    Map<AttackTypes, Double> attacks;
10
11   public Attack() {
12     attacks = new HashMap<>();
13   }
14   @Override
15   public double getDamage(AttackTypes type) {
16     Double damage = attacks.get(type);
17     return (damage != null)? damage.doubleValue():0;
18   }
19   @Override
20   public void setDamage(AttackTypes type, double value) {
21     attacks.put(type, value);
22   }
```

Implementacja

Listing: IUnit.java

```
1  package mediator;
2
3  public interface IUnit {
4
5    public double getProperty(UnitProperties prop);
6    public IAttack getAttack();
7    public void acceptAttack(IAttack attack);
8
9  }
```

Implementacja

Listing: AUnit.java

```
1  package mediator;
2
3  import java.util.HashMap;
4  import java.util.Map;
5
6  public abstract class AUnit implements IUnit {
7
8    protected Map<UnitProperties,Double> stats;
9    protected String name;
10   protected IAttack attack;
11
12   public AUnit(String name) {
13     this.name=name;
14     stats = new HashMap<>();
15     attack = new Attack();
16   }
```

Politechnika Wrocławska

Implementacja

Listing: AUnit.java

```
17    @Override
18    public double getProperty(UnitProperties prop) {
19      Double propVal = stats.get(prop);
20      if(propVal != null)
21        return propVal.doubleValue();
22      return 0;
23    }
24    @Override
25    public IAttack getAttack() {return attack;}
26
27  }
```

Implementacja

Listing: Berserker.java

```
1  package mediator;
2
3  public class Berserker extends AUnit {
4    public Berserker(String name) {
5      super(name);
6      this.stats.put(UnitProperties.HP, 100.0);
7      this.stats.put(UnitProperties.BERSERK, 100.0);
8      this.attack.setDamage(AttackTypes.PHYSICAL, 10);
9    }
10   @Override
11   public void acceptAttack(IAttack attack) {
12     Double pDmg = attack.getDamage(AttackTypes.PHYSICAL);
13     Double currHP = this.stats.get(UnitProperties.HP);
14     if(pDmg !=null) {
15       currHP-= pDmg;
16     }
17     Double fDmg = attack.getDamage(AttackTypes.FIRE);
18     if(fDmg !=null) {
19       currHP-= 0.5*fDmg;
20     }
21     this.stats.put(UnitProperties.HP, currHP<0? 0:currHP );
22   }
```

Implementacja

Listing: FireMage.java

```
1  package mediator;
2  public class FireMage extends AUnit {
3    public FireMage(String name) {
4      super(name);
5      this.stats.put(UnitProperties.HP, 50.0);
6      this.stats.put(UnitProperties.INITIATIVE, 30.0);
7      this.attack.setDamage(AttackTypes.FIRE, 30);
8    }
9    @Override
10   public void acceptAttack(IAttack attack) {
11     Double pDmg = attack.getDamage(AttackTypes.PHYSICAL);
12     Double currHP = this.stats.get(UnitProperties.HP);
13     if(pDmg !=null) {
14       currHP-= 0.9*pDmg;
15     }
16     Double fDmg = attack.getDamage(AttackTypes.FIRE);
17     if(fDmg !=null) {
18       currHP-= 0.1*fDmg;
19     }
20     this.stats.put(UnitProperties.HP, currHP<0? 0:currHP );
21   }
22 }
```

Politechnika Wrocławska

Implementacja

Listing: IDuel.java

```
1 package mediator;
2
3 public interface IDuel {
4   public void setUnits(IUnit unitA, IUnit unitB);
5   public void performDuel();
6 }
```

Implementacja

Listing: ADuel.java

```java
package mediator;

public abstract class ADuel implements IDuel {
  protected IUnit unitA;
  protected IUnit unitB;
  @Override
  public void setUnits(IUnit unitA, IUnit unitB) {this.unitA = unitA; this.unitB = unitB; }
}
```

Implementacja

Listing: DuelSimple.java

```
1  package mediator;
2  public class DuelSimple extends ADuel {
3    @Override
4    public void performDuel() {
5      IUnit first;
6      IUnit second;
7      Double aInit = unitA.getProperty(UnitProperties.INITIATIVE);
8      Double bInit = unitB.getProperty(UnitProperties.INITIATIVE);
9      if(aInit>bInit) {
10       first = unitA;
11       second = unitB;
12     }else {
13       first = unitB;
14       second = unitA;
15     }
16     do {
17       second.acceptAttack(first.getAttack());
18       first.acceptAttack(second.getAttack());
19
20     }while(first.getProperty(UnitProperties.HP)>0 &&
21         second.getProperty(UnitProperties.HP)>0);
22   }
```

Politechnika Wrocławska

Wzorce Projektowe

dr inż. Paweł Trajdos

Politechnika Wrocławska, Katedra Systemów i Sieci Komputerowych
Wyb. Wyspianskiego 27, 50-370 Wroclaw

5 lutego 2023