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

Politechnika Wrocławska

Subsection 1

Observer

Idea

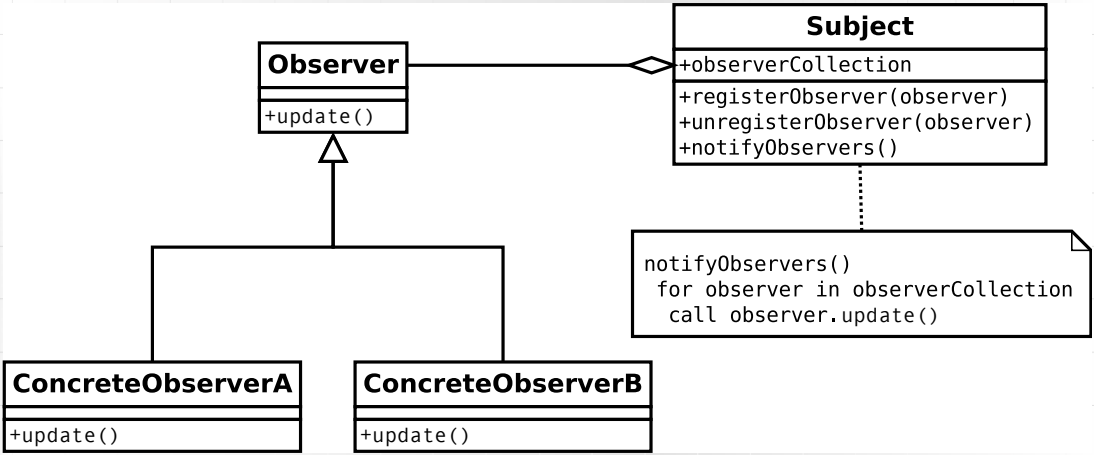
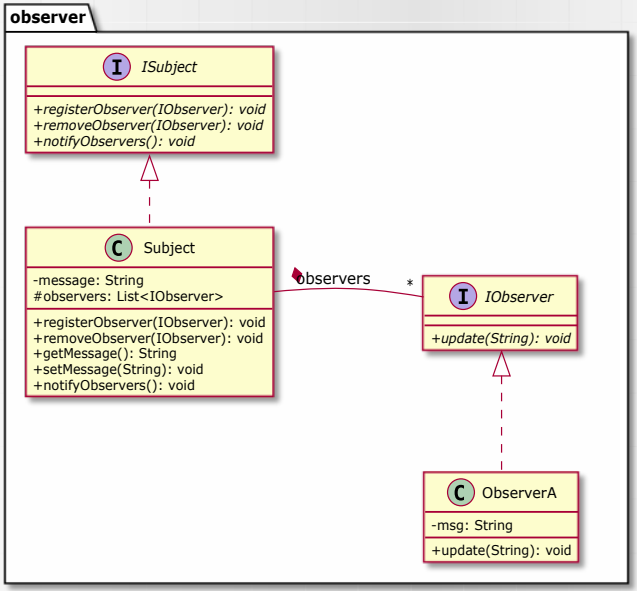

Observer
+update()
Subject
+observerCollection
+registerObserver(observer)
+unregisterObserver(observer)
+notifyObservers()
notifyObservers()
for observer in observerCollection
call observer.update()
ConcreteObserverA
+update()
ConcreteObserverB
+update()

Diagram klas



Politechnika Wrocławska

Diagram obiektów

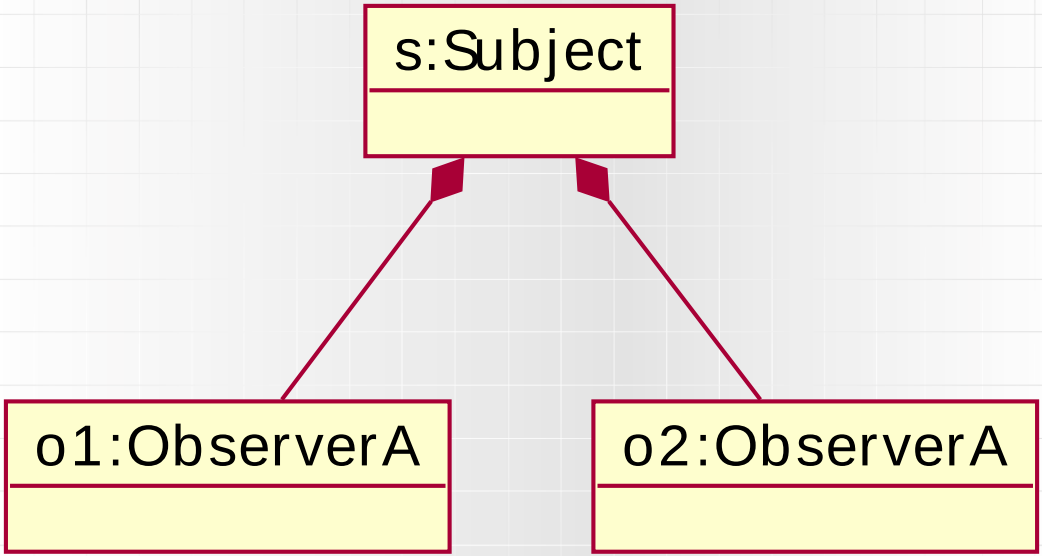
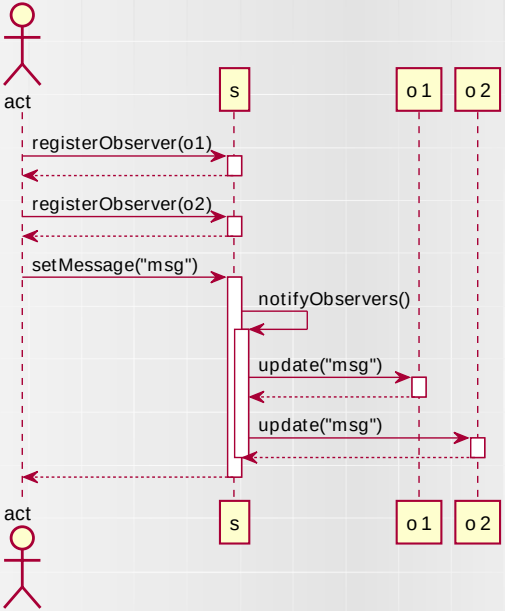


Diagram Sekwencji



Politechnika Wrocławska

Implementacja

Listing: ISubject.java

```
1  package observer;
2
3  public interface ISubject {
4
5    public void registerObserver(IObserver observer);
6    public void removeObserver(IObserver observer);
7    public void notifyObservers();
8
9  }
```

Politechnika Wrocławska

Implementacja

Listing: IObserver.java

```
1 package observer;
2
3 public interface IObserver {
4
5   public void update(String message);
6
7 }
```

Implementacja

Listing: Subject.java

```java
package observer;
import java.util.LinkedList;
import java.util.List;
public class Subject implements ISubject {
  private String message="";
  protected List<IObserver> observers;
  public Subject() {
    observers = new LinkedList<IObserver>();
  }
  @Override
  public void registerObserver(IObserver observer) {
    observers.add(observer);
  }
  @Override
  public void removeObserver(IObserver observer) {
    observers.remove(observer);
  }
  public String getMessage() {return message; }
  public void setMessage(String message) {
    this.message = message;
    this.notifyObservers();
  }
```

Implementacja

Listing: Subject.java

```
23    @Override
24    public void notifyObservers() {
25      for (IObserver iObserver : observers) {
26        iObserver.update(getMessage());
27      }
28    }
29  }
```

Politechnika Wrocławska

Implementacja

Listing: ObserverA.java

```
1 package observer;
2
3 public class ObserverA implements IObserver {
4   private String msg;
5   @Override
6   public void update(String message) {msg=message;}
7 }
```

Politechnika Wrocławska

Subsection 2

Iterator

Idea

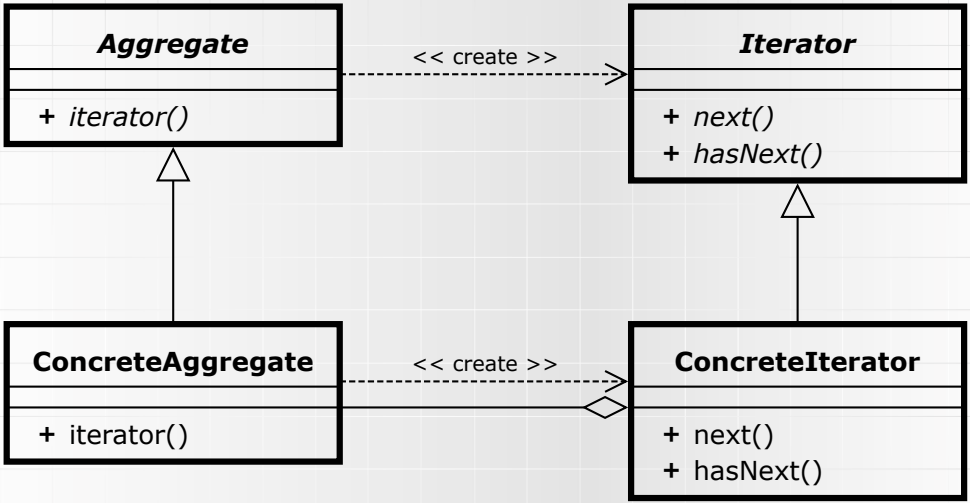

Aggregate
+ iterator()
<< create >>
Iterator
+ next()
+ hasNext()
ConcreteAggregate
+ iterator()
<< create >>
ConcreteIterator
+ next()
+ hasNext()

Diagram klas

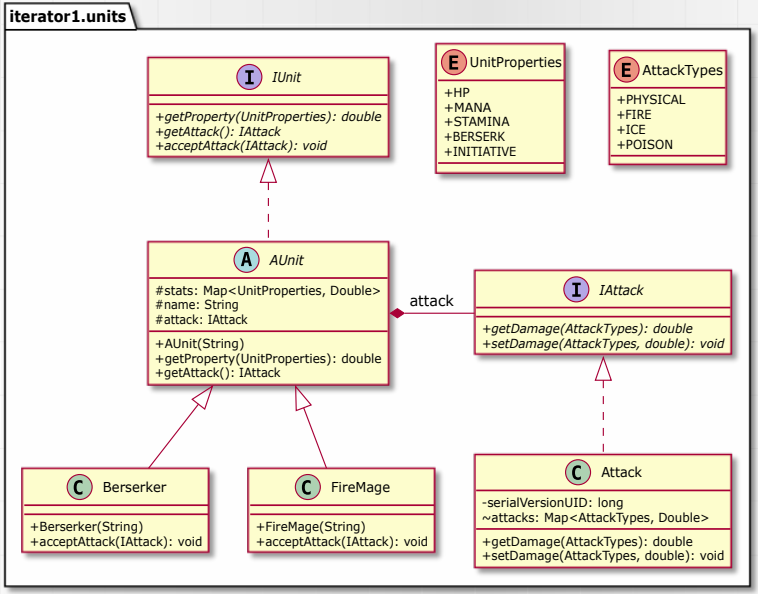
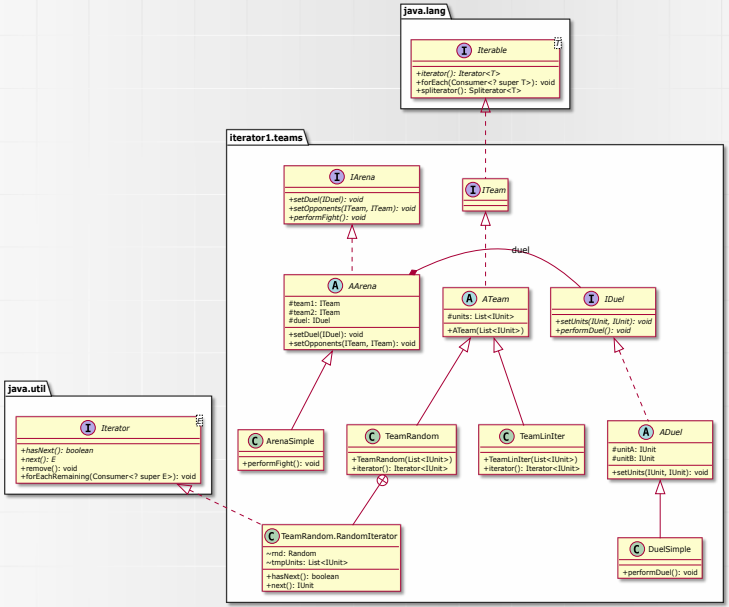


Diagram klas



Politechnika Wrocławska

Implementacja

Listing: IArena.java

```
1  package iterator1.teams;
2
3  public interface IArena {
4
5    public void setDuel(IDuel duel);
6
7    public void setOpponents(ITeam team1, ITeam team2);
8
9    public void performFight();
10
11 }
```

Implementacja

Listing: AArena.java

```
1  package iterator1.teams;
2
3  public abstract class AArena implements IArena {
4    protected ITeam team1;
5    protected ITeam team2;
6    protected IDuel duel;
7    @Override
8    public void setDuel(IDuel duel) {
9      this.duel = duel;
10   }
11
12   @Override
13   public void setOpponents(ITeam team1, ITeam team2) {
14     this.team1 = team1; this.team2 = team2;
15   }
16 }
```

Politechnika Wrocławska

Implementacja

Listing: ArenaSimple.java

```java
package iterator1.teams;

import java.util.Iterator;

import iterator1.units.IUnit;

public class ArenaSimple extends AArena {

  @Override
  public void performFight() {

    Iterator<IUnit> team1Units = this.team1.iterator();
    Iterator<IUnit> team2Units = this.team2.iterator();
    while(team1Units.hasNext() && team2Units.hasNext()) {
      this.duel.setUnits(team1Units.next(), team2Units.next());
      this.duel.performDuel();
    }
  }
}
```

Politechnika Wrocławska

Implementacja

Listing: ITeam.java

```java
package iterator1.teams;

import iterator1.units.IUnit;

public interface ITeam extends Iterable<IUnit> {

}
```

Implementacja

Listing: ATeam.java

```java
package iterator1.teams;

import java.util.LinkedList;
import java.util.List;

import iterator1.units.IUnit;

public abstract class ATeam implements ITeam {

  protected List<IUnit> units;

  public ATeam(List<IUnit> units) {
    this.units = new LinkedList<IUnit>();
    this.units.addAll(units);
  }
}
```

Implementacja

Listing: TeamLinIter.java

```java
package iterator1.teams;

import java.util.Iterator;
import java.util.List;

import iterator1.units.IUnit;

public class TeamLinIter extends ATeam {

  public TeamLinIter(List<IUnit> units) {
    super(units);
  }

  @Override
  public Iterator<IUnit> iterator() {
    return this.units.iterator();
  }

}
```

Politechnika Wrocławska

Implementacja

Listing: TeamRandom.java

```
1  package iterator1.teams;
2
3  import java.util.Iterator;
4  import java.util.LinkedList;
5  import java.util.List;
6  import java.util.Random;
7
8  import iterator1.units.IUnit;
9
10 public class TeamRandom extends ATeam {
11
12   public TeamRandom(List<IUnit> units) {
13     super(units);
14   }
15
16   private class RandomIterator implements Iterator<IUnit>{
```

Implementacja

Listing: TeamRandom.java

```
16  private class RandomIterator implements Iterator<IUnit>{
17
18    Random rnd = new Random();
19    List<IUnit> tmpUnits;
20
21    public RandomIterator() {
22      this.tmpUnits = new LinkedList<IUnit>();
23      tmpUnits.addAll(units);
24    }
25    @Override
26    public boolean hasNext() {
27      return !tmpUnits.isEmpty();
28    }
29
30    @Override
31    public IUnit next() {
32      int rndIndex = this.rnd.nextInt(tmpUnits.size());
33      IUnit retUnit = this.tmpUnits.get(rndIndex);
34      this.tmpUnits.remove(rndIndex);
35      return retUnit;
36    }
37  }
```

Implementacja

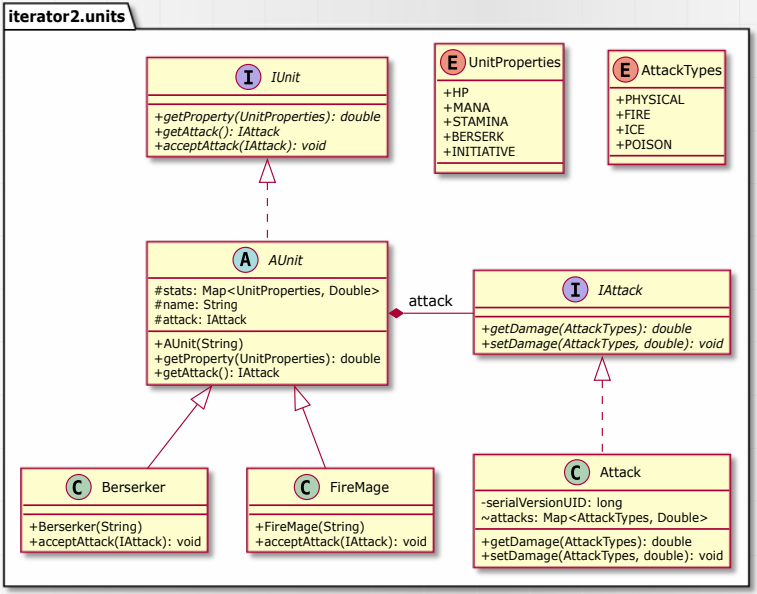
Listing: TeamRandom.java

```
39    @Override
40    public Iterator<IUnit> iterator() {
41      return new RandomIterator();
42    }
43
44  }
```

Politechnika Wrocławska

Alternatywny projekt

Diagram klas



Politechnika Wrocławska

Diagram klas

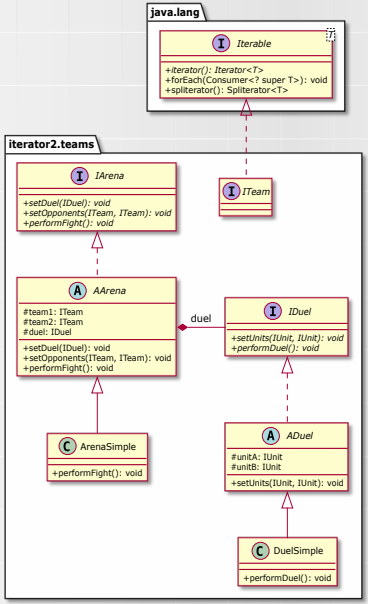
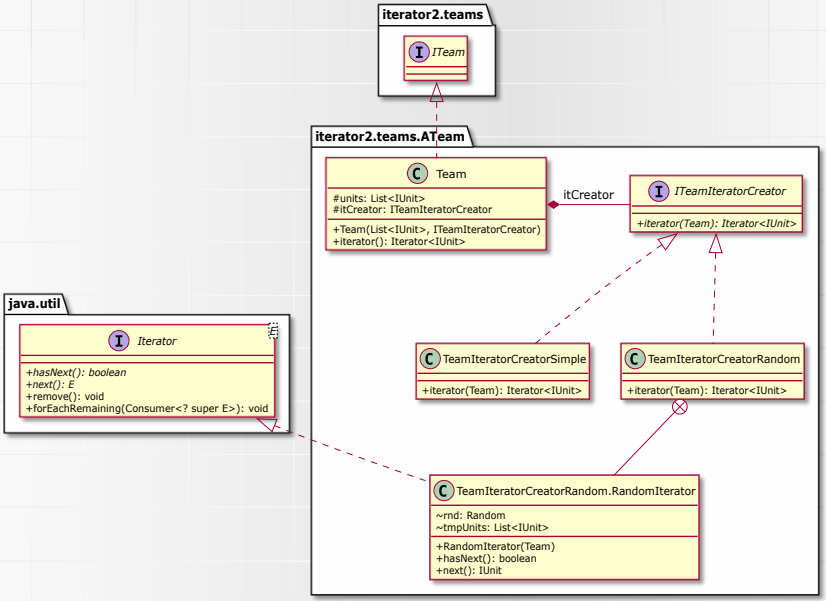


Diagram klas



Listing: Team.java

```java
package iterator2.teams.ATeam;

import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import iterator2.teams.ITeam;
import iterator2.units.IUnit;

public class Team implements ITeam {

  protected List<IUnit> units;
  protected ITeamIteratorCreator itCreator;

  public Team(List<IUnit> units, ITeamIteratorCreator iterCreator) {
    this.units = new LinkedList<IUnit>();
    this.units.addAll(units);
    this.itCreator = iterCreator;
  }

  @Override
  public Iterator<IUnit> iterator() {
    return this.itCreator.iterator(this);
  }
}
```

Politechnika Wrocławska

Listing: ITeamIteratorCreator.java

```java
package iterator2.teams.ATeam;

import java.util.Iterator;

import iterator2.units.IUnit;

public interface ITeamIteratorCreator {

  public Iterator<IUnit> iterator(Team team);

}
```

Politechnika Wrocławska

Listing: TeamIteratorCreatorSimple.java

```java
package iterator2.teams.ATeam;

import java.util.Iterator;

import iterator2.units.IUnit;

public class TeamIteratorCreatorSimple implements ITeamIteratorCreator {


  @Override
  public Iterator<IUnit> iterator(Team team) {
    return team.units.iterator();
  }

}
```

Politechnika Wrocławska

Listing: TeamIteratorCreatorRandom.java

```
1  package iterator2.teams.ATeam;
2
3  import java.util.Iterator;
4  import java.util.LinkedList;
5  import java.util.List;
6  import java.util.Random;
7
8  import iterator2.units.IUnit;
9
10 public class TeamIteratorCreatorRandom implements ITeamIteratorCreator {
11
12   private class RandomIterator implements Iterator<IUnit>{
```

Politechnika Wrocławska

Listing: TeamIteratorCreatorRandom.java

```
12  private class RandomIterator implements Iterator<IUnit>{
13
14    Random rnd = new Random();
15    List<IUnit> tmpUnits;
16
17    public RandomIterator(Team team) {
18      this.tmpUnits = new LinkedList<IUnit>();
19      tmpUnits.addAll(team.units);
20    }
21    @Override
22    public boolean hasNext() {
23      return !tmpUnits.isEmpty();
24    }
25
26    @Override
27    public IUnit next() {
28      int rndIndex = this.rnd.nextInt(tmpUnits.size());
29      IUnit retUnit = this.tmpUnits.get(rndIndex);
30      this.tmpUnits.remove(rndIndex);
31      return retUnit;
32    }
33  }
```

Politechnika Wrocławska

Listing: TeamIteratorCreatorRandom.java

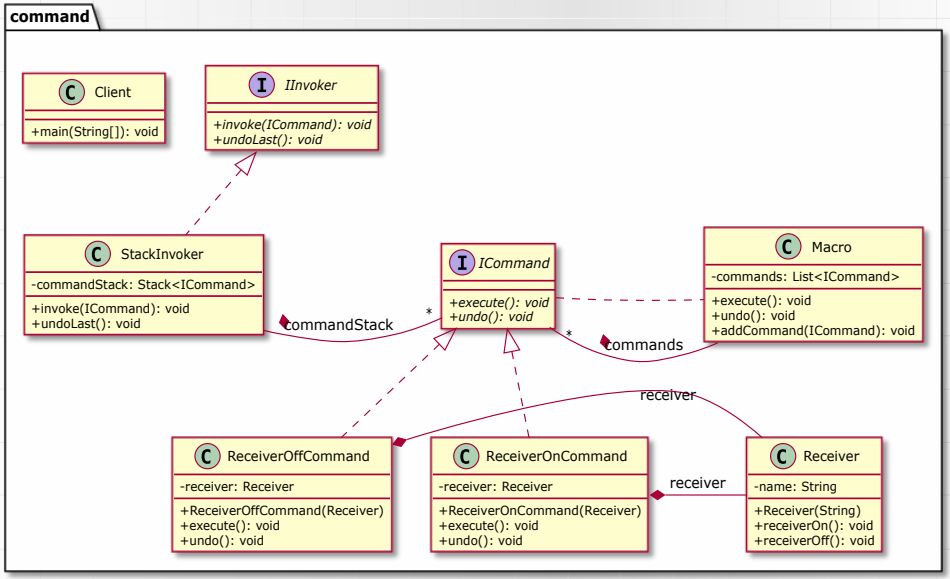
```
35   @Override
36   public Iterator<IUnit> iterator(Team team) {
37     return new RandomIterator(team);
38   }
39
40 }
```

Politechnika Wrocławska

Subsection 3

Command

Diagram klas



Politechnika Wrocławska

Implementacja

Listing: ICommand.java

```
1  package command;
2
3  public interface ICommand {
4
5    public void execute();
6    public void undo();
7
8  }
```

Implementacja

Listing: Macro.java

```java
package command;

import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;

public class Macro implements ICommand {

  private List<ICommand> commands = new LinkedList<>();

  @Override
  public void execute() {
    for (ICommand iCommand : commands)
      iCommand.execute();
  }
  @Override
  public void undo() {
    ListIterator<ICommand> commandIter = commands.listIterator(commands.size());
    while(commandIter.hasPrevious())
      commandIter.previous().undo();
  }
```

Politechnika Wrocławska

Implementacja

Listing: Macro.java

```
23    public void addCommand(ICommand command) {
24      commands.add(command);
25    }
26
27  }
```

Implementacja

Listing: Receiver.java

```java
package command;

public class Receiver {

  private String name;

  public Receiver(String name) {this.name = name;}

  public void receiverOn() {
    System.out.println(name + " On");
  }

  public void receiverOff() {
    System.out.println(name + " Off");
  }

  @Override
  public String toString() {
    return "Receiver: " + name;
  }

}
```

Implementacja

Listing: ReceiverOnCommand.java

```
1  package command;
2
3  public class ReceiverOnCommand implements ICommand {
4
5    private Receiver receiver;
6
7    public ReceiverOnCommand(Receiver receiver) {this.receiver = receiver;}
8    @Override
9    public void execute() {
10     receiver.receiverOn();
11   }
12   @Override
13   public void undo() {
14     receiver.receiverOff();
15   }
16
17 }
```

Implementacja

Listing: ReceiverOffCommand.java

```java
package command;

public class ReceiverOffCommand implements ICommand {

  private Receiver receiver;

  public ReceiverOffCommand(Receiver receiver) {this.receiver = receiver;}
  @Override
  public void execute() {
    receiver.receiverOff();
  }
  @Override
  public void undo() {
    receiver.receiverOn();
  }

}
```

Implementacja

Listing: IInvoker.java

```
1  package command;
2
3  public interface IInvoker {
4
5    public void invoke(ICommand command);
6
7    public void undoLast();
8
9  }
```

Implementacja

Listing: StackInvoker.java

```java
package command;

import java.util.Stack;

public class StackInvoker implements IInvoker {

  private Stack<ICommand> commandStack = new Stack<>();

  @Override
  public void invoke(ICommand command) {
    command.execute();
    commandStack.push(command);
  }

  @Override
  public void undoLast() {
    if(!commandStack.isEmpty())
      commandStack.pop().undo();
  }
}
```

Implementacja

Listing: Client.java

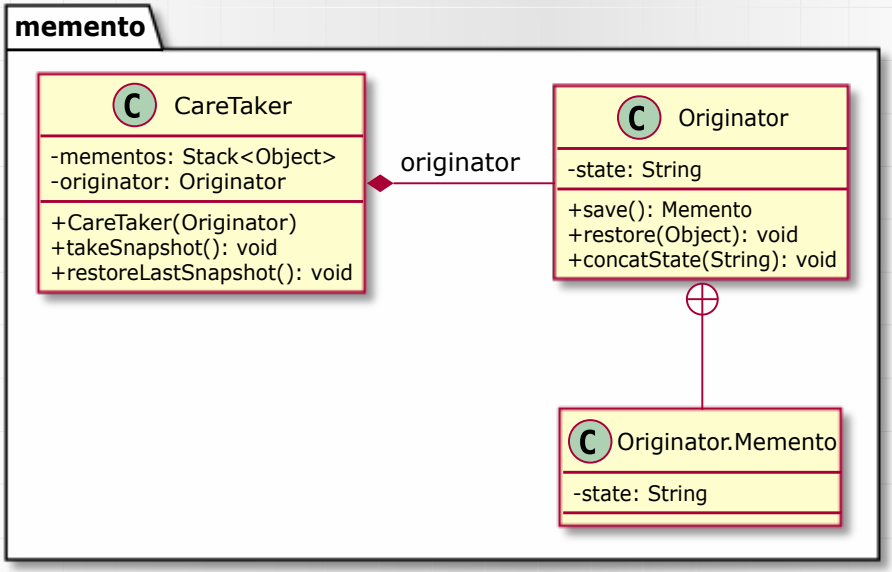
```
1  package command;
2
3  public class Client {
4
5
6    public static void main(String[] args) {
7      Receiver receiver1 = new Receiver("Receiver 1");
8      Receiver receiver2 = new Receiver("Receiver 2");
9      IInvoker invoker = new StackInvoker();
10
11     Macro macro = new Macro();
12     macro.addCommand(new ReceiverOnCommand(receiver1));
13     macro.addCommand(new ReceiverOnCommand(receiver2));
14
15     invoker.invoke(macro);
16
17     invoker.undoLast();
18
19   }
20
21 }
```

Politechnika Wrocławska

Subsection 4

Memento

Diagram klas



Implementacja

Listing: Originator.java

```
1  package memento;
2  public class Originator {
3    private String state="state";
4
5    private class Memento{
6      private String state;
7      private Memento() {
8        this.state = new String(Originator.this.state);
9      }
10   }
11
12   public Memento save() {   return new Memento(); }
13
14   public void restore(Object memento) {
15     this.state = new String(((Memento)memento).state);
16   }
17
18   public void concatState(String toConcat) {
19     this.state.concat(toConcat);
20   }
21 }
```

Politechnika Wrocławska

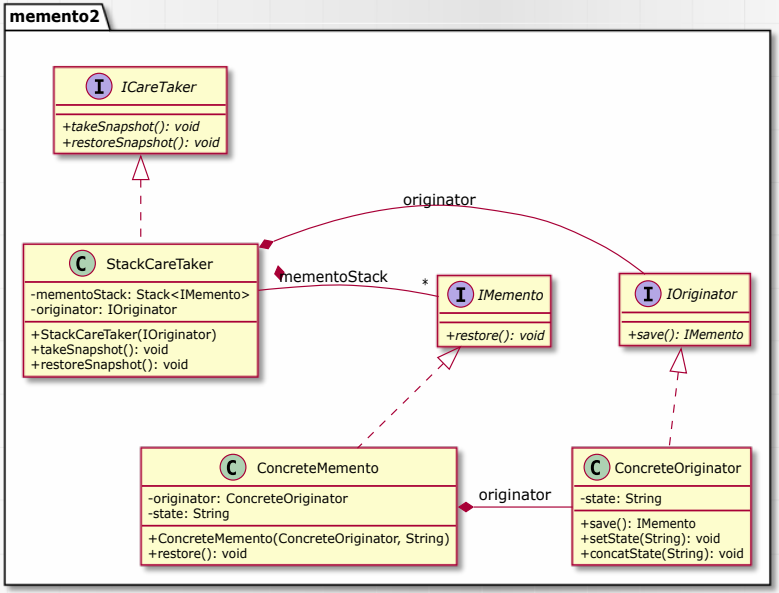
Implementacja

Listing: CareTaker.java

```java
package memento;

import java.util.Stack;

public class CareTaker {

  private Stack<Object> mementos = new Stack<>();
  private Originator originator;

  public CareTaker(Originator originator) {this.originator = originator;}

  public void takeSnapshot() {
    this.mementos.push(originator.save());
  }

  public void restoreLastSnapshot() {
    this.originator.restore(mementos.pop());
  }
}
```

Alternatywne podejście

Diagram klas



Implementacja

Listing: IOriginator.java

```
1 package memento2;
2
3 public interface IOriginator {
4     public IMemento save();
5 }
```

Implementacja

Listing: IMemento.java

```
1  package memento2;
2
3  public interface IMemento {
4
5    public void restore();
6
7  }
```

Implementacja

Listing: ICareTaker.java

```
1 package memento2;
2
3 public interface ICareTaker {
4
5   public void takeSnapshot();
6
7   public void restoreSnapshot();
8 }
```

Implementacja

Listing: ConcreteOriginator.java

```
1  package memento2;
2
3  public class ConcreteOriginator implements IOriginator {
4
5    private String state="state";
6
7
8    @Override
9    public IMemento save() {
10     return new ConcreteMemento(this, state);
11   }
12
13   public void setState(String state) {
14     this.state = new String(state);
15   }
16
17   public void concatState(String toConcat) {
18     this.state.concat(toConcat);
19   }
20
21 }
```

Implementacja

Listing: ConcreteMemento.java

```
1  package memento2;
2
3  public class ConcreteMemento implements IMemento {
4
5    private ConcreteOriginator originator;
6    private String state;
7
8    public ConcreteMemento(ConcreteOriginator originator, String state) {
9      this.originator = originator;
10     this.state = new String(state);
11   }
12
13   @Override
14   public void restore() {
15     this.originator.setState(state);
16   }
17
18 }
```

Implementacja

Listing: StackCareTaker.java

```java
package memento2;

import java.util.Stack;

public class StackCareTaker implements ICareTaker {

  private Stack<IMemento> mementoStack = new Stack<>();
  private IOriginator originator;

  public StackCareTaker(IOriginator originator) { this.originator = originator;}
  @Override
  public void takeSnapshot() {
    this.mementoStack.push(originator.save());
  }

  @Override
  public void restoreSnapshot() {
    mementoStack.pop().restore();
  }

}
```

Politechnika Wrocławska

Wzorce Projektowe

dr inż. Paweł Trajdos

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

5 lutego 2023