Politechnika Wrocławska

Prosta symulacja

dr inż. Paweł Trajdos

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

5 lutego 2023

Politechnika Wrocławska

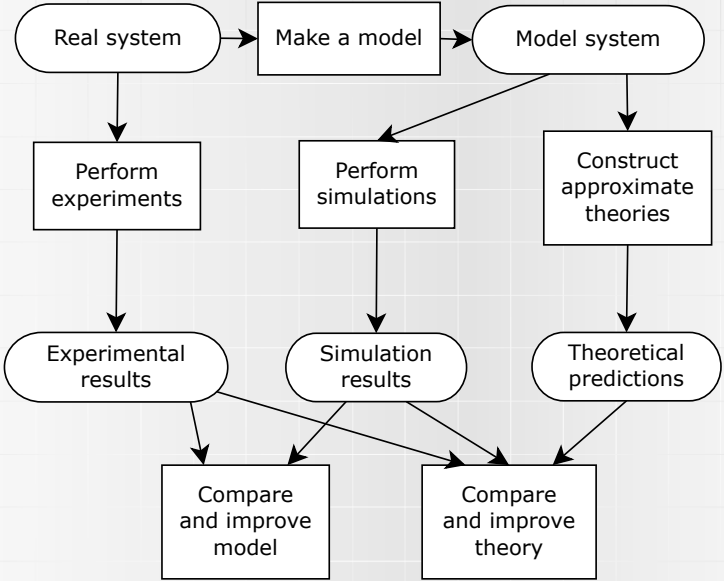
Spis treści

Politechnika Wrocławska

Section 1

Symulacje komputerowe

Po co nam symulacje?



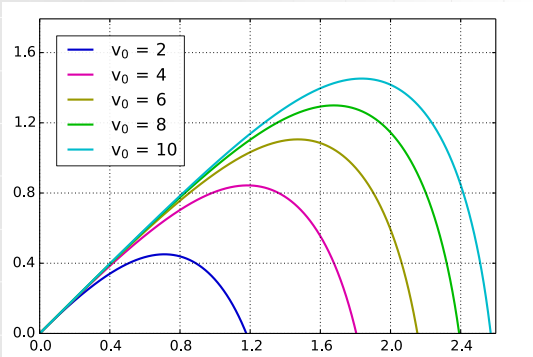
Politechnika Wrocławska

Rodzaje symulacji

- Przewidywalność:
  - deterministyczne,
  - stochastyczne.
- Upływ czasu:
  - czas ciągły,
  - czas dyskretny,
  - symulacja zdarzeń dyskretnych
- Sposób symulacji:
  - oparty o modele analityczne,
  - agent-based.

Rodzaje symulacji

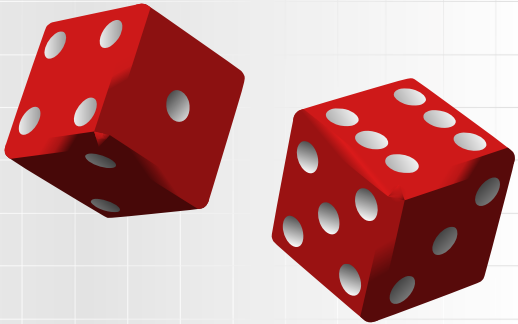
- Przewidywalność:
  - deterministyczne,
  - stochastyczne.
- Upływ czasu:
  - czas ciągły,
  - czas dyskretny,
  - symulacja zdarzeń dyskretnych
- Sposób symulacji:
  - oparty o modele analityczne,
  - agent-based.



Politechnika Wrocławska

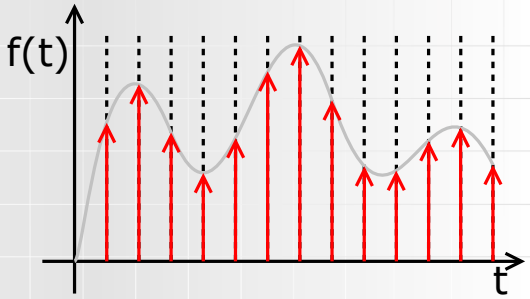
Rodzaje symulacji

- Przewidywalność:
  - deterministyczne,
  - stochastyczne.
- Upływ czasu:
  - czas ciągły,
  - czas dyskretny,
  - symulacja zdarzeń dyskretnych
- Sposób symulacji:
  - oparty o modele analityczne,
  - agent-based.



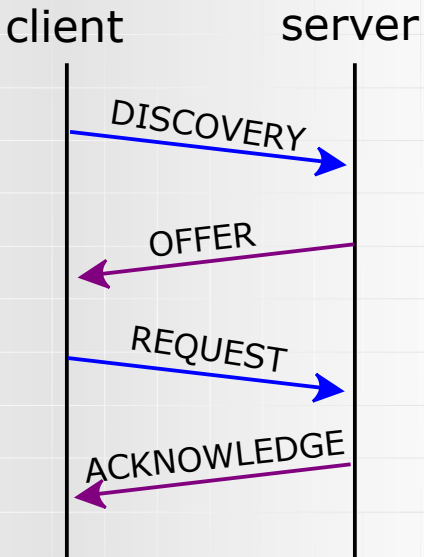
Rodzaje symulacji

- Przewidywalność:
  - deterministyczne,
  - stochastyczne.
- Upływ czasu:
  - czas ciągły,
  - czas dyskretny,
  - symulacja zdarzeń dyskretnych
- Sposób symulacji:
  - oparty o modele analityczne,
  - agent-based.



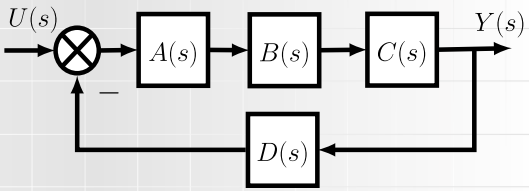
Rodzaje symulacji

- Przewidywalność:
  - deterministyczne,
  - stochastyczne.
- Upływ czasu:
  - czas ciągły,
  - czas dyskretny,
  - symulacja zdarzeń dyskretnych
- Sposób symulacji:
  - oparty o modele analityczne,
  - agent-based.



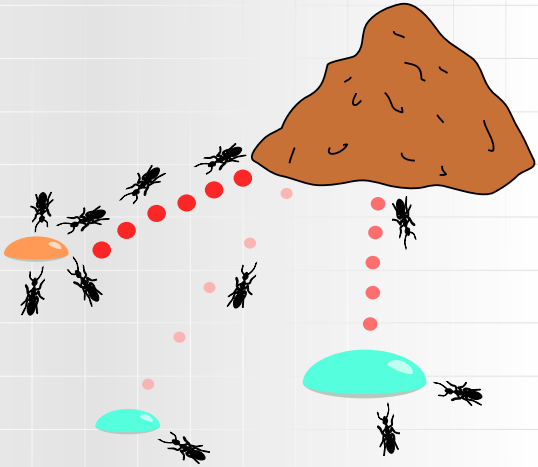
Rodzaje symulacji

- Przewidywalność:
  - deterministyczne,
  - stochastyczne.
- Upływ czasu:
  - czas ciągły,
  - czas dyskretny,
  - symulacja zdarzeń dyskretnych
- Sposób symulacji:
  - oparty o modele analityczne,
  - agent-based.

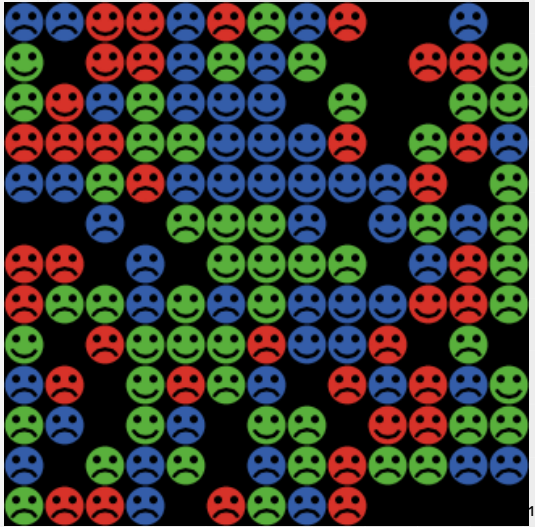


Rodzaje symulacji

- Przewidywalność:
  - deterministyczne,
  - stochastyczne.
- Upływ czasu:
  - czas ciągły,
  - czas dyskretny,
  - symulacja zdarzeń dyskretnych
- Sposób symulacji:
  - oparty o modele analityczne,
  - agent-based.



Symulacje agentowe

 [1]

[1] D. Robertson. *Eight Mile and the Emergence of Segregation.* 2019. URL: http://www.duncanrobertson.com/2017/03/09/eight-mile-emergence-segregation/ (term. wiz. 02. 04. 2020).

Politechnika Wrocławska

Section 2

Projektujemy prostą symulację

Politechnika Wrocławska

Analiza czasownikowo - rzeczownikowa

Projektujamy prostą symulację agentową, w której będziemy badać zachowanie osobników należących do różnych grup. Dla uproszczenia przyjmujemy, że osobniki będą zamieszkiwać jednowymiarową przestrzeń o zadanej wielkości. Przestrzeń podzielona będzie na sektory, a każdy z tych sektorów może być zamieszkany przez jednego osobnika.

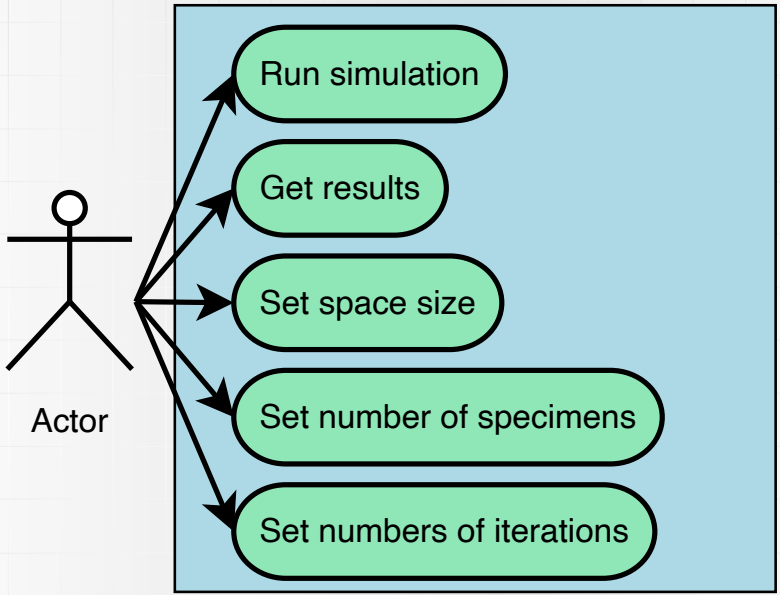
- Zachowanie się osobników:
  - Każdy z osobników będzie dążył do tego by osiąść w okolicy zamieszkanej przez jego ziomków (osobniki należące do tej samej grupy).
  - Osobnik zdecyduje się na zamieszkanie w danym polu jeżeli w jego sąsiedztwie będzie więcej jego ziomków niż obcych (osobniki z innych grup).
- Parametry symulacji:
  - Wielkość przestrzeni S.
  - Liczba osobników każdej z grup K_i, $\sum K_i < S$.
  - Maksymalna liczba iteracji I

Analiza czasownikowo - rzeczownikowa

Projektujamy prostą symulację agentową, w której będziemy badać zachowanie osobników należących do różnych grup. Dla uproszczenia przyjmujemy, że osobniki będą zamieszkiwać jednowymiarową przestrzeń o zadanej wielkości. Przestrzeń podzielona będzie na sektory, a każdy z tych sektorów może być zamieszkany przez jednego osobnika.

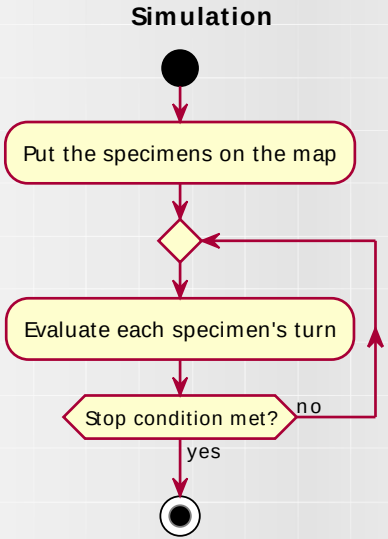
- Zachowanie się osobników:
  - Każdy z osobników będzie dążył do tego by osiąść w okolicy zamieszkanej przez jego ziomków (osobniki należące do tej samej grupy).
  - Osobnik zdecyduje się na zamieszkanie w danym polu jeżeli w jego sąsiedztwie będzie więcej jego ziomków niż obcych (osobniki z innych grup).
- Parametry symulacji:
  - Wielkość przestrzeni S.
  - Liczba osobników każdej z grup K_i, $\sum K_i < S$.
  - Maksymalna liczba iteracji I

Diagram UC

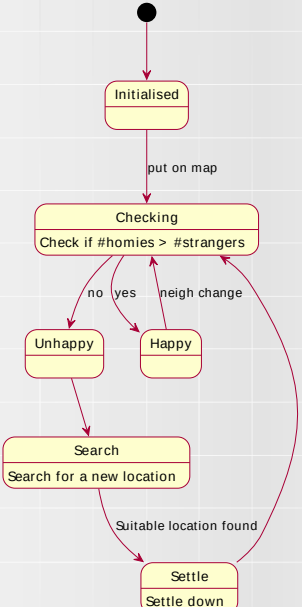

Run simulation
Get results
Set space size
Set number of specimens
Set numbers of iterations
Actor

Symulacja

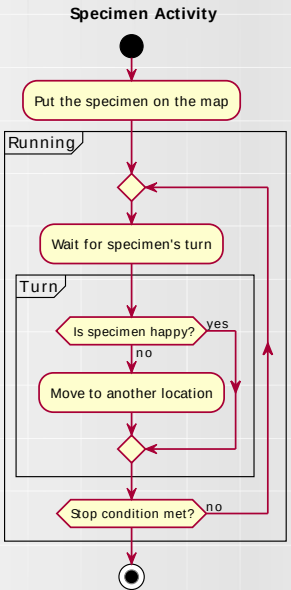
Diagram aktywności



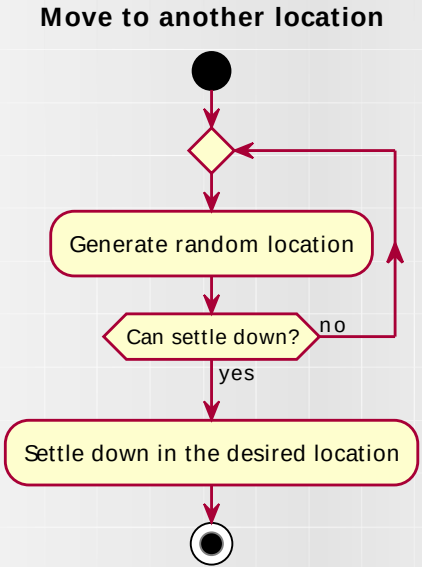
Osobnik – diagram stanów



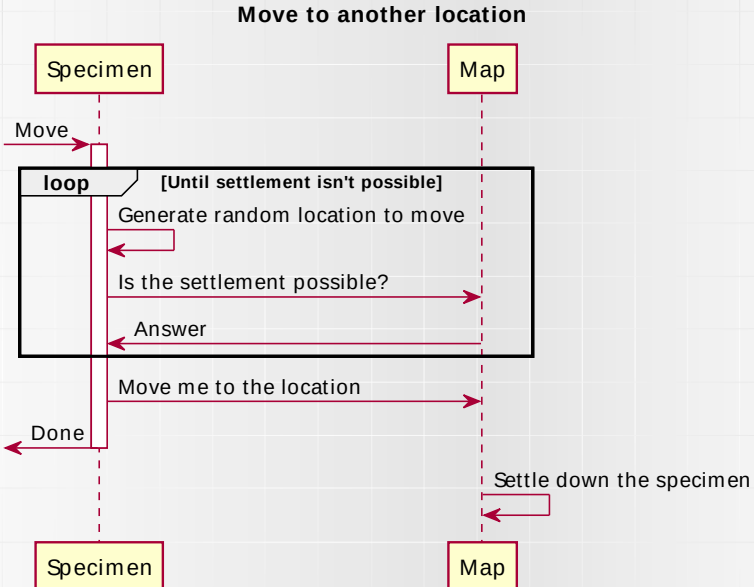
Osobnik – diagram aktywności



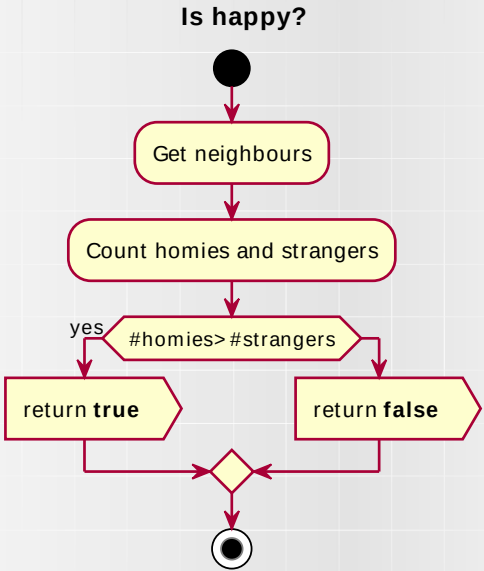
Osobnik – Przeniesienie do innej lokacji



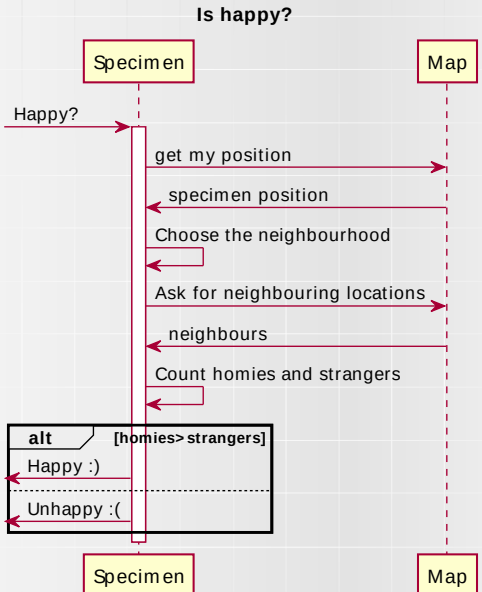
Osobnik – Przeniesienie do innej lokacji



Osobnik – Sprawdzenie czy jest szczęśliwy



Osobnik – Sprawdzenie czy jest szczęśliwy



Karty CRC

Classname: Simulation	
Superclass: none **Subclass(es):** none	
Responisibilities: - Runs the simulation - Read configuration	**Collaboration:** Specimen, Map

Politechnika Wrocławska

Karty CRC

Classname: Specimen

Superclass: none
Subclass(es): Other specimens

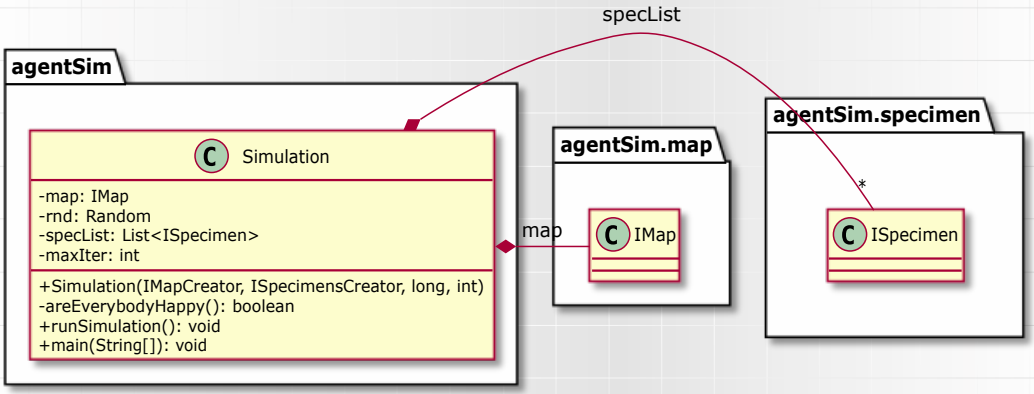
Responisibilities:
Handle Specimen specific operations.

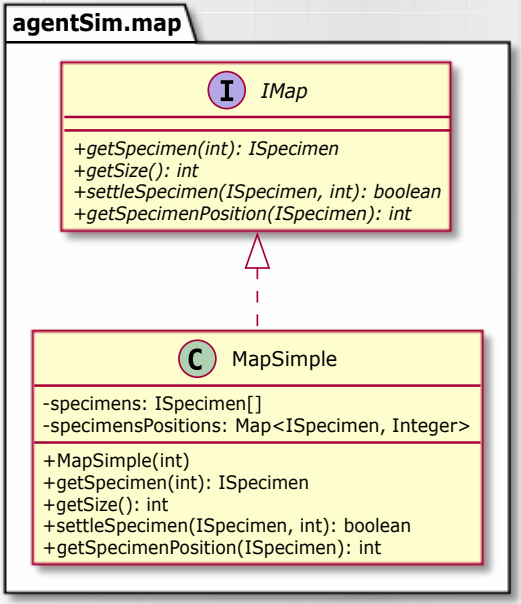
- Determine if h appy
- Find neighbours
- Change location

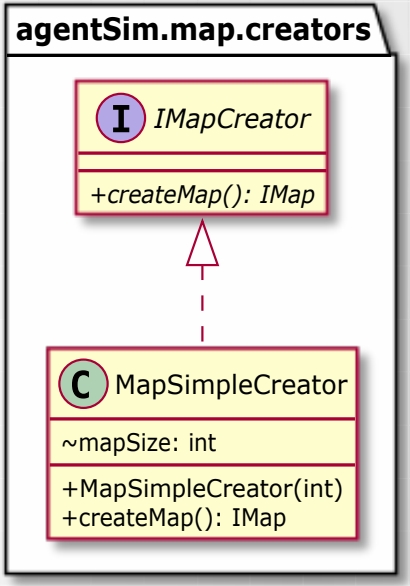
Collaboration:
Map

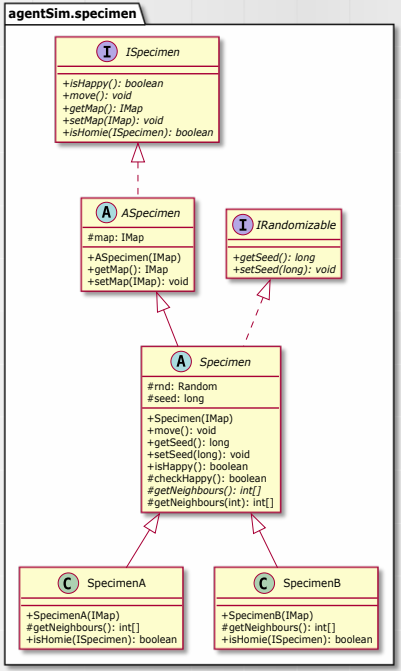
Karty CRC

Classname: Map	
Superclass: none **Subclass(es):** none	
Responisibilities: Stores specimens	**Collaboration:** Specimen


specList
agentSim
Simulation
-map: IMap
-rnd: Random
-specList: List<ISpecimen>
-maxIter: int
+Simulation(IMapCreator, ISpecimensCreator, long, int)
-areEverybodyHappy(): boolean
+runSimulation(): void
+main(String[]): void
map
agentSim.map
IMap
agentSim.specimen
*
ISpecimen


agentSim.map
IMap
+getSpecimen(int): ISpecimen
+getSize(): int
+settleSpecimen(ISpecimen, int): boolean
+getSpecimenPosition(ISpecimen): int
MapSimple
-specimens: ISpecimen[]
-specimensPositions: Map<ISpecimen, Integer>
+MapSimple(int)
+getSpecimen(int): ISpecimen
+getSize(): int
+settleSpecimen(ISpecimen, int): boolean
+getSpecimenPosition(ISpecimen): int


agentSim.map.creators
I
IMapCreator
+createMap(): IMap
C
MapSimpleCreator
~mapSize: int
+MapSimpleCreator(int)
+createMap(): IMap


agentSim.specimen
ISpecimen
+isHappy(): boolean
+move(): void
+getMap(): IMap
+setMap(IMap): void
+isHomie(ISpecimen): boolean
ASpecimen
#map: IMap
+ASpecimen(IMap)
+getMap(): IMap
+setMap(IMap): void
IRandomizable
+getSeed(): long
+setSeed(long): void
Specimen
#rnd: Random
#seed: long
+Specimen(IMap)
+move(): void
+getSeed(): long
+setSeed(long): void
+isHappy(): boolean
#checkHappy(): boolean
#getNeighbours(): int[]
#getNeighbours(int): int[]
SpecimenA
+SpecimenA(IMap)
#getNeighbours(): int[]
+isHomie(ISpecimen): boolean
SpecimenB
+SpecimenB(IMap)
#getNeighbours(): int[]
+isHomie(ISpecimen): boolean

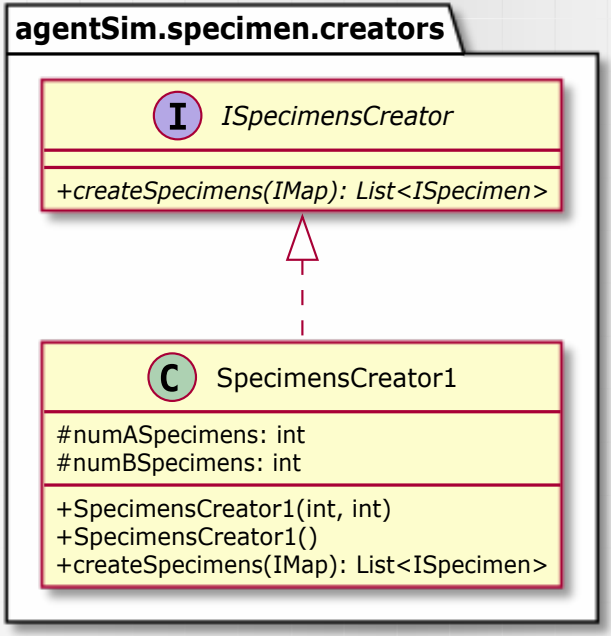
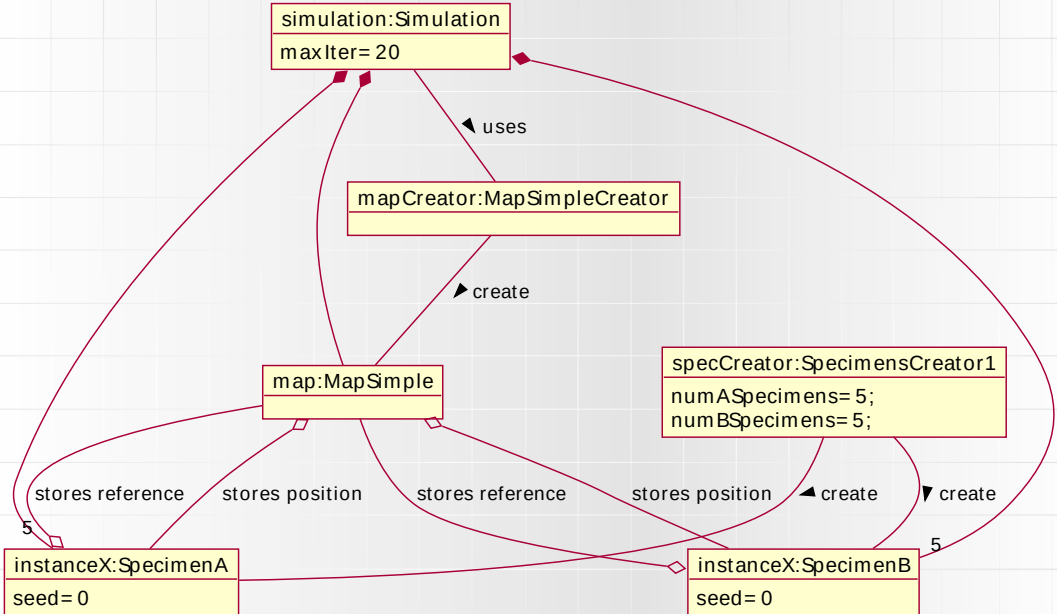

agentSim.specimen.creators
I
ISpecimensCreator
+createSpecimens(IMap): List<ISpecimen>
C
SpecimensCreator1
#numASpecimens: int
#numBSpecimens: int
+SpecimensCreator1(int, int)
+SpecimensCreator1()
+createSpecimens(IMap): List<ISpecimen>

Diagram obiektów



Politechnika Wrocławska

Implementacja

Listing: ISpecimen.java

```java
package agentSim.specimen;

import agentSim.map.IMap;

public interface ISpecimen {

  public boolean isHappy();
  public void move();
  public IMap getMap();
  public void setMap(IMap map);
  public boolean isHomie(ISpecimen spec);

}
```

Implementacja

Listing: ASpecimen.java

```java
package agentSim.specimen;

import agentSim.map.IMap;

public abstract class ASpecimen implements ISpecimen {

  protected IMap map;

  public ASpecimen(IMap map) {this.map = map; }
  @Override
  public IMap getMap() {return map;}
  @Override
  public void setMap(IMap map) {this.map=map;}
}
```

Implementacja

Listing: IRandomizable.java

```java
package agentSim.specimen;

public interface IRandomizable {

  public long getSeed();
  public void setSeed(long seed);

}
```

Implementacja

Listing: Specimen.java

```
1  package agentSim.specimen;
2  import java.util.Random;
3
4  import agentSim.map.IMap;
5
6  public abstract class Specimen extends ASpecimen implements IRandomizable {
7
8    protected Random rnd;
9    protected long seed=0;
10
11   public Specimen(IMap map) {super(map);  rnd = new Random(seed);}
12
13   @Override
14   public void move() {
15     if(checkHappy())return;
16     do {
17       int position = rnd.nextInt(map.getSize());
18       if(map.getSpecimen(position) == null) {
19         map.settleSpecimen(this, position);
20         break;
21       }
22     }while(true);
```

Implementacja

Listing: Specimen.java

```
25  @Override
26  public long getSeed() {return this.seed;}
27
28  @Override
29  public void setSeed(long seed) { this.seed=seed; rnd.setSeed(seed);   }
30
31  @Override
32  public boolean isHappy() {return this.checkHappy();}
```

Implementacja

Listing: Specimen.java

```
34  protected boolean checkHappy() {
35    int neighbours[] = getNeighbours();
36
37    if(neighbours == null) return false;
38
39    int numHomies=0;
40
41    for(int i=0;i<neighbours.length;i++)
42      if(this.isHomie(map.getSpecimen(neighbours[i])))
43        numHomies++;
44
45    double perc = ((double)numHomies)/neighbours.length;
46
47    if(perc>=0.5)return true;
48
49    return false;
50  }
```

Implementacja

Listing: Specimen.java

```
52  protected abstract int[] getNeighbours();
53
54  protected int[] getNeighbours(int size){
55    int tmp;
56    int currPos =map.getSpecimenPosition(this);
57    tmp =  currPos - size;
58    int minPos= tmp>=0? tmp:0;
59    tmp= currPos + size;
60
61    int maxPos = tmp>=map.getSize()? (map.getSize()-1):tmp;
62
63    int[] neighs = new int[maxPos - minPos];
64    int cnt=0;
65
66    for(int i=minPos;i<=maxPos;i++) {
67      if(i==currPos)continue;
68      neighs[cnt++]=i;
69    }
70    return neighs;
71  }
```

Implementacja

Listing: SpecimenA.java

```java
package agentSim.specimen;

import agentSim.map.IMap;

public class SpecimenA extends Specimen {

  public SpecimenA(IMap map) { super(map);}
  @Override
  public String toString() {return isHappy()? "A":"a";}
  @Override
  protected int[] getNeighbours() {return getNeighbours(2);}
  @Override
  public boolean isHomie(ISpecimen spec) {
    if(spec instanceof SpecimenA)return true;
    return false;
  }

}
```

Implementacja

Listing: SpecimenB.java

```java
package agentSim.specimen;

import agentSim.map.IMap;

public class SpecimenB extends Specimen {

  public SpecimenB(IMap map) {super(map); }
  @Override
  public String toString() {return isHappy()? "B":"b";}
  @Override
  protected int[] getNeighbours() {return getNeighbours(3);}
  @Override
  public boolean isHomie(ISpecimen spec) {
    if(spec instanceof SpecimenB)return true;
    return false;
  }
}
```

Politechnika Wrocławska

Implementacja

Listing: ISpecimensCreator.java

```java
package agentSim.specimen.creators;

import java.util.List;

import agentSim.map.IMap;
import agentSim.specimen.ISpecimen;

public interface ISpecimensCreator {

  public List<ISpecimen> createSpecimens(IMap map);

}
```

Implementacja

Listing: SpecimensCreator1.java

```java
package agentSim.specimen.creators;

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

import agentSim.map.IMap;
import agentSim.specimen.ISpecimen;
import agentSim.specimen.SpecimenA;
import agentSim.specimen.SpecimenB;

public class SpecimensCreator1 implements ISpecimensCreator {

  protected int numASpecimens;
  protected int numBSpecimens;

  public SpecimensCreator1(int numASpecimens,int numBSpecimens) {
    this.numASpecimens = numASpecimens;
    this.numBSpecimens = numBSpecimens;
  }
  public SpecimensCreator1() {
    this(10,10);
  }
```

Implementacja

Listing: SpecimensCreator1.java

```
24    @Override
25    public List<ISpecimen> createSpecimens(IMap map) {
26      List<ISpecimen> specList = new LinkedList<>();
27
28      for(int i=0;i<numASpecimens;i++)
29        specList.add(new SpecimenA(map));
30
31      for(int i=0;i<numBSpecimens;i++)
32        specList.add(new SpecimenB(map));
33
34      return specList;
35    }
36
37  }
```

Politechnika Wrocławska

Implementacja

Listing: IMap.java

```java
package agentSim.map;

import agentSim.specimen.ISpecimen;

public interface IMap {

  public ISpecimen getSpecimen(int position);
  public int getSize();
  public boolean settleSpecimen(ISpecimen spec, int position);
  public int getSpecimenPosition(ISpecimen spec);

}
```

Implementacja

Listing: MapSimple.java

```java
package agentSim.map;

import java.util.HashMap;
import java.util.Map;

import agentSim.specimen.ISpecimen;

public class MapSimple implements IMap {

  private ISpecimen[] specimens;
  private Map<ISpecimen, Integer> specimensPositions;

  public MapSimple(int size) {
    specimens = new ISpecimen[size];
    specimensPositions = new HashMap<>();
  }

  @Override
  public ISpecimen getSpecimen(int position) {return specimens[position];}

  @Override
  public int getSize() {return specimens.length;  }
```

Implementacja

Listing: MapSimple.java

```
24  @Override
25  public boolean settleSpecimen(ISpecimen spec,int position) {
26    int settled = getSpecimenPosition(spec);
27    if(specimens[position]!= null)
28      return false;
29    if(settled >=0)
30      specimens[settled]=null;
31    spec.setMap(this);
32    specimens[position] = spec;
33    specimensPositions.put(spec, position);
34    return true;
35  }
36
37  @Override
38  public int getSpecimenPosition(ISpecimen spec) {
39     Integer pos = specimensPositions.get(spec);
40     if(pos == null)
41       return -1;
42     return pos.intValue();
43  }
```

Implementacja

Listing: MapSimple.java

```
45  @Override
46  public String toString() {
47    StringBuffer buff = new StringBuffer();
48    for(int i=0;i<specimens.length;i++) {
49      if(specimens[i] == null) {buff.append("#");}
50      else {buff.append(specimens[i].toString());}
51    }
52    return buff.toString();
53  }
```

Implementacja

Listing: IMapCreator.java

```
1  package agentSim.map.creators;
2
3  import agentSim.map.IMap;
4
5  public interface IMapCreator {
6
7    public IMap createMap();
8
9  }
```

Implementacja

Listing: MapSimpleCreator.java

```java
package agentSim.map.creators;

import agentSim.map.IMap;
import agentSim.map.MapSimple;

public class MapSimpleCreator implements IMapCreator {

  int mapSize;
  public MapSimpleCreator(int mapSize) {
    this.mapSize = mapSize;
  }

  @Override
  public IMap createMap() {
    return new MapSimple(mapSize);
  }

}
```

Implementacja

Listing: Simulation.java

```
13 public class Simulation {
14
15   private IMap map;
16   private Random rnd;
17   private List<ISpecimen> specList;
18   private int maxIter;
19
20   public Simulation(IMapCreator mapCreator,ISpecimensCreator specCreator,long seed, int maxIter)
        {
21
22     map =mapCreator.createMap();
23
24     rnd = new Random(seed);
25     specList = specCreator.createSpecimens(map);
26
27     for(int i=0;i<specList.size();i++)
28       while(!map.settleSpecimen( specList.get(i), rnd.nextInt(map.getSize()) ) );
29     this.maxIter = maxIter;
30   }
```

Implementacja

Listing: Simulation.java

```
32  private boolean areEverybodyHappy() {
33    for (ISpecimen iSpecimen : specList)
34      if(! iSpecimen.isHappy())return false;
35    return true;
36  }
37
38  public  void runSimulation() {
39    int iters = maxIter;
40    System.out.println(map.toString());
41    do {
42      if(areEverybodyHappy())break;
43      for (ISpecimen iSpecimen : specList) {
44        iSpecimen.move();
45        System.out.println(map.toString());
46      }
47    }while(--iters>0);
48  }
```

Implementacja

Listing: Simulation.java

```
50  public static void main(String[] args) {
51
52    MapSimpleCreator mapCreat = new MapSimpleCreator(20);
53    ISpecimensCreator specCreat = new SpecimensCreator1(5,5);
54
55    Simulation sim  = new Simulation(mapCreat, specCreat,1, 20);
56    sim.runSimulation();
57    System.out.println("END");
58  }
```

Politechnika Wrocławska

Implementacja

```
1 ....
2 BBBBB#######AAAA##a#
3 BBBBB#######AAAA##a#
4 BBBBB#######AAAA##a#
5 BBBBB#######AAAA##a#
6 BBBBB######AAAAA####
7 BBBBB######AAAAA####
```

Prosta symulacja

dr inż. Paweł Trajdos

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

5 lutego 2023