Szoftvertechnológia

Házi feladat

Tóth Péter

NEPTUN

BME IIT, 2016

# Feladatleírás

[A feladat max. 1000 karakteres részletezettségű magyar nyelvű leírása.]

# Követelmények

### Funkcionális követelmények

[Az alábbi táblázat kitöltésével készítendő.]

|  |  |  |  |
| --- | --- | --- | --- |
| **Azonosító** | **Leírás** | **Use-case** | **Komment** |
|  |  |  |  |

### Egyéb követelmények

[Nem funkcionális követelmények, korlátozások, a táblázat kitöltésével]

|  |  |  |
| --- | --- | --- |
| **Azonosító** | **Leírás** | **Komment** |
|  |  |  |

# Use-case leírás

## Use-case leírások

[Minden use-case-hez külön. 3-5 use-case. Legyen 1 use-case, amit legalább egy másik use-case include-ol!]

|  |  |
| --- | --- |
| **Use-case neve** | Timer tick event |
| **Rövid leírás** |  |
| **Aktorok** | Timer |
| **Forgatókönyv** |  |

|  |  |
| --- | --- |
| **Use-case neve** | Truck unload |
| **Rövid leírás** |  |
| **Aktorok** | Truck |
| **Forgatókönyv** |  |

|  |  |
| --- | --- |
| **Use-case neve** | Boss throws a switch |
| **Rövid leírás** |  |
| **Aktorok** | Boss |
| **Forgatókönyv** |  |

|  |  |
| --- | --- |
| **Use-case neve** | Boss changes the speed of a conveyor |
| **Rövid leírás** |  |
| **Aktorok** | Boss |
| **Forgatókönyv** |  |

|  |  |
| --- | --- |
| **Use-case neve** |  |
| **Rövid leírás** |  |
| **Aktorok** |  |
| **Forgatókönyv** |  |

|  |  |
| --- | --- |
| **Use-case neve** |  |
| **Rövid leírás** |  |
| **Aktorok** |  |
| **Forgatókönyv** |  |

## Use-case diagram

# Statikus leírás

[Minden, a feladatban szereplő osztály (interfész) rövid, egy bekezdés hosszú ismertetése. Az osztályok száma 10 és 20 között legyen.]

interface **HasInput**:

Box fogadásához szükséges műveleteket definiálja.

**insert(Box):** Fogadja a paraméterben kapott csomagot.

interface **HasOutput**:

Box átadásához szükséges műveleteket definiálja.

**pop(Box)**: **Box**-ot továbbít a rácsatolt **hasInput** felé.

abstract class **TransportObject** implements **HasInput**, **HasOutput**:

Általános csomagátviteli feladatokat implementál. Alapértelmezetten a **Ground**-ra fog esni a csomag. A legtöbb esetben ebből érdemes származtatni csomagátvitellel foglalkozó objektumokat..

class Color:

meghatároz egy színt

class Event:

Egy eseményt határoz meg, tárol egy EvType és egy EvData mezőt

enum EvType:

meghatározza az Event típusát, azaz, hogy a data referenciáját milyen típussá kell castolni, hogy hozzáférjünk

abstract class EventData:

egy esemény részletei, különböző események adatstruktúrái ebből származnak

interface Observable:

Event-eket képes generálni Observer pattern szerint.

fireEvent(Event): meghívja a regisztrált Observerek onEvent(Event) függvényét

interface Observer:

képes az Observable által generált Event-ek elkapására és feldolgozására

onEvent(Event): akkor hívódik, amikor számára érdekes Event keletkezik. Ha több fajta Event-et figyel egyszerre, akkor itt érdemes megvizsgálni az EvType értéket, azonosítani az adattípust, amivé castolni kell a kapott adatmezőben szereplő EventData-t és aszerint meghívni az azt feldolgozó függvényt.

class Timer implements Observable:

időzítőt valósít meg, adott időközönként generál egy Tick event-et, ami az idő múlását reprezentálja. Bármi az az idő folytával változtatja az állapotát erre kell feliratkozzon.

class TimeData extends EventData:

A Tick event adatszerkezete, tárolja az event keletkezésekori időt és a legutóbbi Tick óta eltelt időt.

class Ground implements hasInput:

Alapértelmezetten ő fogad Box-okat a kimenetekről. Ha fogad egy Box-ot, megsemmisíti.

class Box implements Observable:

A központban közlekedő csomag megvalósítása. Tárol egy Color és egy Time (expiry) mezőt.

A rendszer Timer objektumára feliratkozik keletkezéskor.

destroy(): generál egy BoxDestroyed event-et. Akkor hívódik, ha egy Tick event értéke meghaladja az expiry értékét.

getColor(): Lehetővé teszi a doboz színének vizsgálatát

onTick(): Tick event vételekor hívódik. Ez dolgozza fel.

class LogisticsIn extends TransportObject:

Beérkező Truck-ok képesek csatlakozni rá és csomagokat juttatni a rendszerbe.

class LogisticsOut extends TransportObject implements Observable:

Kimenő Truck-ok képesek csatlakozni rá és "felpakolni" róla. Tárol egy szín értéket, ami a helyes müködés esetében várt csomagszínt határozza meg. Ha a szűrőtől eltérő színű csomagot kap, generál egy BoxAtInvalidLocation eventet.

class BoxAtInvalidLocData implements EventData:

A BoxAtInvalidLocation event részletei. A kérdéses Box referenciáját tárolja.

class Truck extends TransportObject:

A kimenő és bejövő teherautókat valósítja meg.

Box fogadása esetén generál egy BoxLoaded event-et.

startUnpacking(): Megkezdi a csomagok bevitelét a csatlakoztatott LogisticsIn felé.

class BoxLoadedData implements EventData:

A BoxLoaded event részletei. A kérdéses Box referenciáját tárolja.

class ConveyorBelt extends TransportObject:

Futószalagot valósít meg. Létrehozáskor elkezdi figyelni a Timert. Időre lépteti a rajta lévő Box-okat. Amikor egy Box a futószalag végére ér, akkor továbbítja a kimenetén.

Megjegyzés: egy Box képes megsemmisülni, amíg a futószalagon tartózkodik, ha egy másik Box ráesik.

setSpeed(float): A futószalag sebességét lehet vele állítani, akár teljesen leállítható.

onTick(): Tick event vételekor hívódik. Ez dolgozza fel.

class ConveyorSlot extends TransportObject:

class Switch extends TransportObject:

class SwitchOutput extends TransportObject:

class EfficiencyEvaluator:

interface Modifier:

class Reward implements Modifier:

class Penalty implements Modifier:

## Osztályok leírása

### Osztály1

#### Felelősség

[Mi az osztály felelőssége. Kb 1 bekezdés.]

#### Metódusok

[Milyen publikus metódusokkal rendelkezik. Metódusonként egy mondat arról, hogy a metódus mit csinál.]

* **int foo(Osztály3 o1, Osztály4 o2)**: metódus leírása
* **int bar(Osztály5 o1)**: metódus leírása

### Osztály2

#### Felelősség

[Mi az osztály felelőssége. Kb 1 bekezdés.]

#### Metódusok

[Milyen publikus metódusokkal rendelkezik. Metódusonként egy mondat arról, hogy a metódus mit csinál.]

* **int foo(Osztály3 o1, Osztály4 o2)**: metódus leírása
* **int bar(Osztály5 o1)**: metódus leírása

## Osztálydiagram

# Dinamika leírása

[Inicializálásra, use-case-ekre. Konzisztens kell legyen az előző ponttal. Minden metódus, ami ott szerepel, fel kell tűnjön valamelyik szekvenciában. Minden metódusnak, ami szekvenciában szerepel, szerepelnie kell a valamelyik osztálydiagramon.]

## [Use-case neve1] szekvenciája

## [Use-case neve2] szekvenciája

# Állapotmodell

[Csak azokhoz az osztályokhoz, ahol van értelme. Egyetlen állapotból álló state-chartok ne szerepeljenek. Legalább egy osztályhoz tartozzon state-chart.]

## [Osztály1] állapotgépe

# Tevékenységnapló

[A referencia a leírásban szereplő tevékenységgel érintett alpontot jelöli.]

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Leírás** | **Referencia** |
| 2015.11.13. 13:00 | 2 óra | Class Diagram tervezése (félkész)  WhiteStarUML-ről megállapítás, hogy egy hulladék | szofttech\_hf.uml |
| 2015.11.21. 00:00 | 1 óra | ClassDiagram kiegészítés, javítgatás | szofttech\_hf.uml |
| 2015.11.21. 16:00 | 3 óra | ClassDiagram  befejezés, ellenőrzés, rendezés | szofttech\_hf.uml |
| 2015.11.22. 1:10 | 3 | Class leírások elkészítése  Feladat számára lényegtelen függvények kukázása | szofttech\_hf.uml  2. feladat, |
| 2015.11.22. 10:30 | 2 | Class diagramból pár, a |  |
|  |  |  |  |
|  |  |  |  |

# Egyéb

A felhasznált modellező eszköz: *WhiteStarUML-lel kezdve, utána StarUML*

Egyéb felhasznált eszköz és a használat célja: *XXXXXX*

A feladat megoldására fordított teljes munkaidő órában: *XX*