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

**Alap funkcionális építőelemek:**

interface **HasInput**:

**Box** fogadásához szükséges műveleteket definiálja.

**insert(Box):** Fogadja a paraméterben kapott **Box**-ot.

abstract class **HasOutput**:

Box átadásához szükséges műveleteket definiálja. Eltárolja a rákötött **HasInput** referenciáját.

**pop(Box)**: **Box**-ot továbbít a rácsatolt **hasInput** felé. Ha nincs rákötve input, megsemmisíti a **Box**-ot.

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

Általános csomagátviteli feladatokat implementál. A legtöbb esetben ebből érdemes származtatni csomagátvitellel foglalkozó objektumokat..

class **Color**:

meghatároz egy színt

class **Time**:

meghatároz egy időpontot / időmennyiséget

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 **Observer**-ek **onEvent(Event)** függvényét

interface **Observer**:

képes az **Observable** által generált **Event**-ek elkapására és feldolgozására

**listenTo(Observable)**: feliratkozik az **Observable** event-jeire.

**ignore(Observable)**: leiratkozik az **Observable** event-jeiről.

**onEvent(Event)**: akkor hívódik, amikor számára érdekes **Event** keletkezik. Ha több fajta **Event**-et figyel egyszerre, akkor ebben 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 múlásával változtatja az állapotát erre kell feliratkozzon.

class **TickData** extends **EventData**:

A **Tick** event adatszerkezete, tárolja az event keletkezésekori időt és a legutóbbi **Tick** óta eltelt időt.**Logisztikai állomás elemei:**

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, ezután törli az event-jeire feliratkozók listáját.

**getColor()**: Lehetővé teszi a doboz színének vizsgálatát

**onTick()**:Ha az időérték meghaladja-e az expiry-t, generál egy **BoxDestroyed** event-et.

class **BoxDestroyedData** implements **EventData**:

A **BoxDestroyed** event részletei. A kérdéses **Box** referenciáját tárolja.

class **LogisticsIn** extends **TransportObject**:

Beérkező **Truck**-ok képesek csatlakozni rá és csomagokat juttatni a rendszerbe. **Truck** csatlakozásakor generál egy **TruckConnected** eventet. (ez nincs a diagramon, mert az összekötés folyamatát nem kell bemutatni)

**insert(Box)**: Generál egy **BoxEntry** eventet.

class **BoxEntryData** implements **EventData**:

A **BoxEntry** event részletei. A kérdéses **Box** referenciáját tárolja.

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. **Truck** csatlakozásakor generál egy **TruckConnected** eventet. (ez nincs a diagramon, mert az összekötés folyamatát nem kell bemutatni)

**insert(Box)**: Ha a szűrőtől eltérő színű csomagot kap, generál egy **BoxAtInvalidExit** eventet.

class **BoxInvalidExitData** implements **EventData**:

A **BoxAtInvalidExit** event részletei. A kérdéses **Box** referenciáját tárolja.

class **Truck** extends **TransportObject**:

A kimenő és bejövő teherautókat valósítja meg.

**insert(Box)**: Feliratkozik a **Box** eventjeire és generál egy **BoxLoaded** event-et.

**startUnpacking()**: Megkezdi a csomagok bevitelét a csatlakoztatott **LogisticsIn** felé és leiratkozik az eventjeikről.

**onBoxDestroyed(BoxDstrData)**: Eltávolítja a megsemmisült **Box**-ot.

class **BoxLoadedData** implements **EventData**:

A **BoxLoaded** event részletei. A kérdéses **Box** referenciáját tárolja.

class **Switch** implements **HasInput**:

Egy elágazást valósít meg. Egy váltóval állítható, merre továbbítsa a rajta átmenő **Box**-okat. Tárol egy bool-t ami a kapcsoló állását jelzi, illetve 2 **SwitchOutput**-ot, ezekre képes csak továbbítani **Box**-okat.

**insert(Box)**: a beállításnak megfelelő output **pop(Box)** függvényét hívja meg.

**setState(bool)**: beállítja a használt kimenetet.

class **SwitchOutput** extends **HasOutput**:

Mivel a Switch alapból nem örökli a **HasOutput** class-t, ez teszi lehetővé, hogy más logisztikai objektumok bemenetei csatolhatóak legyenek rá.

class **ConveyorBelt** extends **TransportObject**:

Futószalagot valósít meg. Létrehozáskor elkezdi figyelni a Timert. Képes léptetni a rajta lévő **Box**-okat. A bemenete a futószalag egyik, a kimenete a futószalag másik végén a mozgásiránynak megfelelően helyezkedik el. Ha egy **Box** elér a futószalag végére, meghívódik a **pop(Box)**.

**insert(Box)**: elhelyezi a **Box**-ot a futószalag elején és feliratkozik az eventjeire. Ütközés esetén megsemmisítve a korábban ott lévőt.

**pop(Box)**: Leiratkozik a **Box** eventjeiről.

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

**onTick(TimeData)**: **Tick** event vételekor hívódik. Az eltelt idő és a sebesség szerint változtatja a pozicióját a rajta utazó **Box**-oknak.

**onBoxDestroyed(BoxDstrData)**: Eltávolítja a megsemmisült **Box**-ot a futószalagról, leiratkozni nem kell, mert a **Box** megszüntette a kapcsolatot.

class **ConveyorSlot** implements **HasInput**:

Egy **ConveyorBelt**-hez csatolható bemenetet valósít meg, amivel a futószalag közepére is lehet beküldeni **Box**-ot. Tárol egy referenciát a cél **ConveyorBelt**-re és egy float értéket, ami a beillesztés pontját határozza meg.

Megjegyzés: mivel nem általános kimenetet használ az objektum, ezért csak a bemenete szabványos és közvetlenül a cél **ConveyorBelt** privát adataihoz hozzáférve helyezi el bele az érkezett **Box**-ot.

**insert()**: Elhelyezi a **Box**-ot a megadott ponton és feliratkozik az eventjeire. Ütközés esetén megsemmisítve a korábban ott lévőt.

class **Boss**:

Vezérli a logisztikai központot.

**setSwitch(Switch, bool)**: A kapott **Switch**-et a paraméterben kapott állapotba állítja.

**setConveyorSpeed(ConveyorBelt, float)**: A kapott **ConveyorBelt**-et a paraméterben kapott sebességre állítja.

class **EfficiencyEvaluator** implements **Observer**:

Ez felel a logisztikai rendszer teljesítményfelméréséért és a **Boss** jutalmazása / büntetéséért. Létrejöttekor feliratkozik az összes **LogisticsIn** és **LogisticsOut** objektum event-jeire.

**onBoxEntry()**: Feliratkozik a kapott **Box** event-jeire.

**onTruckConnected()**: Feliratkozik a kérdéses **Truck** eventjeire.

**onTruckDisconnected()**: Leiratkozik a kérdéses **Truck** eventjeiről.

**onBoxLoaded()**: Jutalomban részesíti a **Boss**-t. Leiratkozik a **Box** eventjeiről.

**onBoxDestroyed()**: Büntetésben részesíti a **Boss**-t.

**onBoxAtInvalidExit()**: Büntetésben részesíti a **Boss**-t. Leiratkozik a **Box** eventjeiről.

interface **Modifier**:

Egy állapotmódosítót valósít meg, ami képes az alanyra valamilyen hatást gyakorolni.

**castEffect(Boss)**: alkalmazza a módosító hatást a paraméterben kapott **Boss**-ra.

class **Reward** implements **Modifier**:

Pozitív hatást alkalmaz egy célponton.

**castEffect(Boss)**: Jutalmat ad a paraméterben kapott **Boss**-nak.

class **Penalty** implements **Modifier**:

Negatív hatást alkalmaz egy célponton.

**castEffect(Boss)**: Büntetést hajt végre a paraméterben kapott **Boss**-on.

## 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. 1:15  2015.11.22. 12:30 | 2  2 | Class-ok összegzése, felsorolása, leírások készítése  Feladat számára lényegtelen függvények kukázása  Apróbb javítások | szofttech\_hf.uml  4. feladat, |
| 2015.11.23. 12:00 | 3 | Class diagram apróságok javítása  Class leírások befejezése, formázása | szofttech\_hf.uml  4. feladat |
|  |  |  |  |
|  |  |  |  |

# 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*