Software Design Document

ARNON LAZERSON
UML Class Diagram

- A Class Diagram is a diagram describing the structure of a system

- Includes
  - Classes
  - Attributes
  - Operations (or methods)
  - Relationships among the classes
  - Constraint Rules and Notes
Class

Describes a set of objects having similar:
- Attributes (status)
- Operations (behavior)
- Relationships with other classes

Attributes and operations may
- have their visibility marked:
  - "+" for public
  - "#" for protected
  - "-" for private
  - "~" for package

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Window</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>size: Size visibility: boolean</td>
</tr>
<tr>
<td>Attributes</td>
<td>display()</td>
</tr>
<tr>
<td>Operations</td>
<td>hide()</td>
</tr>
</tbody>
</table>
Class Relationships

- Association
  - Employee <-> Company

- Aggregation
  - Car -- 2..* Door

- Composition
  - Circle --> Point

- Generalization
  - Circle --> Shape {abstract}

- Realization
  - LinkedList --> List

- Dependency
  - Iterator <--friend-- Vector
# Multiplicity

## Multiplicity Indicators

<table>
<thead>
<tr>
<th>Multiplicity Indicator</th>
<th>Indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exactly one</td>
<td>1</td>
</tr>
<tr>
<td>Zero or more (unlimited)</td>
<td>* (0..*)</td>
</tr>
<tr>
<td>One or more</td>
<td>1..*</td>
</tr>
<tr>
<td>Zero or one (optional association)</td>
<td>0..1</td>
</tr>
<tr>
<td>Specified range</td>
<td>2..4</td>
</tr>
<tr>
<td>Multiple, disjoint ranges</td>
<td>2, 4, 6, 8</td>
</tr>
</tbody>
</table>
ePark PDOM

**Diagram:***

- **Device**
  - control
  - 1..*

- **Supervisor**
  - +MyBrclt
  - 0..1

- **Child**
  - 0..*
  - 1..*

- **Guardian**
  - guard
  - 1..*

- **Account**
  - approvedBy
  - 1

- **CCCompany**
  - 1..*

- **AppUser**
  - 1

- **Bracelet**
  - 0..1

- **eTicket**
  - 0..*

- **Entry**
ePark PDOM
Usage Manager
Sequence Diagram

- External message
- Synchronous Call
- Return
- Self Call
- Asynchronous Call

object A

object B

Object c

constructor
Communication with external entities is done via proxies. Communication with other components is implemented by self methods.

MyAccount: Account
MyTicket: eTicket

1.0 extRequest("new registration")
1.1 setDetails(Details&CCNumber)
1.2 notify(MeasureChild)
1.3 measureChild()
1.4 notify(getDetails)
1.5 send child and CC details()
1.6 validateCC(CCNumber)
1.7 extRequest("Details: ....")
1.8 alt CCisValid
   [not CCOK]
   1.9 storeInDB()
   1.10 CCOK= validateCC(CCNumber): int
   [CCOK]
   1.11 StoreInDB()
   1.12 notify(BrcltReady)
   1.13 "CC rejected"()
1.14 notify(CCRejected)
1.15 "put child on measuring device"()
1.16 readMeasurement()
1.17 produceBracelet()
1.18 notify(childDetails)
1.19 storeInDB()
1.20 notify(measureChild)
1.21 "put child on measuring device"()
1.22 readMeasurement()
1.23 produceBracelet()
1.24 notify(BrcltReady)
1.25 "take bracelet"()
1.26 notify(measureChild)
1.27 "take bracelet"()
1.28 CCisValid

Guardian: Guardian
MyChild: Child
MyBrclt: Bracelet

guardsman: App
CCCompanyProxy
GuardianAppProxy

Including Bracelet and eTicket
Device Control

PDOM::Device
- ID: int
- description: string
- minAge: int
- minHeight: float
- minWeight: float
+ isOpen: boolean
+ isInOrder: boolean
+ isIdle: boolean
- price: int
- myUser: eTicket
+ isEligible(eTicket): boolean
+ openGate()
+ requestEntrance(int): int

DeviceGateProxy
+ braceletDetected(int)
+ openGate()

PDOM::Child
- ID: int
- age: int
- height: int
- weight: int
- name: string
- usingDevice: int
+ getLocation(): int
- retrieveFromDB(int)
+ init(Details)
- storeInDB()

DeviceUser
+ storeInDB()
+ retrieveFromDB(int)
+ setId(int): int
+ tryToEnter(int): int

PDOM::eTicket
+ validUntil: Time
+ startUsing(Device): int
+ isEligible(int): boolean
+ enterDevice()
+ updateTicketInDB()
SOFTWARE DESIGN DOCUMENT

Device Control
Device Repository Manager
User Repository Manager
Usage Manager

Child (Bracelet)
Device (Gate)

1.0 enter device(ID)
1.1 get device data(): DeviceData
1.2
1.3 get eTicket Data(): eTicketData
1.4
1.5 check child's eligibility(): boolean
1.6 open()
1.7 eTicket update (child entered device)

opt [child eligible to enter]
ref eTicket Update
State diagram

- **State**
  - Normal/passive (waiting)
  - Active (processing)

- **Transition**
  - State change due to an event or condition

- **Event**
  - A trigger causing a transition ("flip light switch")

- **Guard**
  - A condition associated to the event ("flip light switch[door closed]")

- **Action**
  - preformed during transition or within a state
  - Entry/ exit/ do
State Machine: Device Class

stm Device State Machine

standby
- setup
- setup completed
- activate

configuring
- entry / retrieveFromDB
- do / loop: setAttributes
- exit / storeInDB

UnBoarding
- entry / openGate
- do / countExits
- last user exited [stopped normally]

Boarding
- entry / disableOperation
- run
- last user exited [stopped normally]

admitting child
- bracelet idicated
- [entrance not approved]
- entry / approveEntrance
- do / openGate
- do / waitForCHildsEntry
- exit / closeGate

running
- entry / enableOperation
- do / operate
- exit / disableOperation

shutdown
- [entrance not approved]
- setup completed

activate

setup completed

shutdown
[stopped abnormally]

activate

setup completed

shutdown
[entrance not approved]

activate

setup completed

shutdown
[stopped normally]