Topic 5

Introduction to UML Diagrams
Objectives

• To introduce UML Diagrams
  • A diagrammatic way of showing the relationships among classes
  • This will help our understanding of the definitions of our collections and the usage of our collections in applications
UML Diagrams

• **Unified Modeling Language (UML)** is a standard notation for object-oriented design
  • Used to *model* object-oriented designs
  • Shows overall design of a solution
    • Shows class specifications
    • Shows how classes interact with each other
  • Diagrams use specific icons and notations
  • It is *language independent*
UML Class Diagram

- A **class** is represented in a UML diagram by a rectangle divided into 3 sections:
  - **name** of the class
  - **attributes** of the class (i.e. the data fields of the class, including variables and constants)
  - **operations** of the class (essentially equivalent to a Java method or a C++ function)
Example: UML Class Diagram

Person

<table>
<thead>
<tr>
<th>firstName</th>
<th>lastName</th>
<th>email</th>
</tr>
</thead>
</table>

| getName() | getEmail() | setEmail() | equals() | toString() |
Example: UML Class Diagram

SocialNetwork

friendList
numFriends
DEFAULT_MAX_FRIENDS
add()
remove()
toString()
Features of UML Class Diagrams

• Attributes and operations may include:
  • *visibility*: public (+) or private (-)
  • *type* of attribute or operation
  • *parameter list* for operations

• Including this information is of the form:

  visibility variable_name: type
  visibility variable_name: type = default_value
  visibility method_name(parameter_list): return_type
  \{property\}
Example: UML Class Diagram

<table>
<thead>
<tr>
<th>SocialNetwork</th>
</tr>
</thead>
<tbody>
<tr>
<td>- friendList: array of Person</td>
</tr>
<tr>
<td>- numFriends: integer</td>
</tr>
<tr>
<td>- DEFAULT_MAX_FRIENDS = 100</td>
</tr>
<tr>
<td>+ add(friend: Person)</td>
</tr>
<tr>
<td>+ remove(friend: Person): boolean</td>
</tr>
<tr>
<td>+ toString(): String</td>
</tr>
</tbody>
</table>
Features of UML Class Diagrams

• Attributes and operations may be left incomplete, and completed as design is developed
Set of UML Class Diagrams

• A set of UML class diagrams shows:
  • The classes used in the system
  • The *relationships* among classes
  • The *constraints* on the connections among classes
Example: UML Diagram for Order Processing

- **Order**
  - dateReceived
  - isPrepaid
  - orderNumber
  - price
  - dispatch()
  - close()

- **Customer**
  - name
  - address
  - getCreditRating()

- **OrderLine**
  - quantity
  - price
  - product

- **CorporateCustomer**
  - contactName
  - creditRating
  - creditLimit
  - salesRep
  - remind()
  - billForMonth()

- **PersonalCustomer**
  - customerCardNumber
Features of Set of UML Diagrams

- **Association between classes:**
  - Represents a relationship between objects of those classes
  - Indicated with a *solid line* between the classes
  - Can be annotated with *cardinality*: indicates a numeric association between classes, such as:
    - one-to-one
    - one-to-many (*..*)
    - many-to-many (*..*)
    - zero-to-many (0..*)
    - zero-to-5 (0..5)
    - etc.
Example: Association Between Classes

<table>
<thead>
<tr>
<th>LibraryCustomer</th>
<th>LibraryItem</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>callNumber</td>
</tr>
<tr>
<td>address</td>
<td>title</td>
</tr>
<tr>
<td>register( )</td>
<td>checkout( )</td>
</tr>
<tr>
<td>deregister( )</td>
<td>return( )</td>
</tr>
</tbody>
</table>

0..* 0..*
Association Between Classes

• What is the Order-Customer relationship in our Order Processing System?

• How would we annotate that a Library Customer can not check out more than 5 library items?
Features of Set of UML Diagrams

• *Usage of another class*:
  • Broken line with an arrow indicates that one class makes use of the other
  ........................................
  
  • Line can be labeled with a message indicating the type of usage
Example: One Class Indicating its Use of Another

<table>
<thead>
<tr>
<th>LibraryCustomer</th>
<th>Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>location</td>
</tr>
<tr>
<td>address</td>
<td>ipAddress</td>
</tr>
<tr>
<td>register( )</td>
<td>logon( )</td>
</tr>
<tr>
<td>deregister( )</td>
<td>logoff( )</td>
</tr>
</tbody>
</table>

Searches online catalogue
Features of Set of UML Diagrams

• **Implementation of an interface:**
  • Indicated by a broken line with an open arrow
  
  \[\text{---} \rightarrow\]

• **UML diagram for an interface** is much like the UML diagram for a class
  • But there are no attributes (why not?)
UML Diagram for StackADT Interface

<<interface>>
StackADT

push( )
pop( )
peek( )
isEmpty( )
size( )
toString( )
UML Diagram for ArrayStack Implementation of StackADT

<table>
<thead>
<tr>
<th>ArrayStack</th>
<th>&lt;&lt;interface&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>stack</td>
<td>StackADT</td>
</tr>
<tr>
<td>top</td>
<td></td>
</tr>
<tr>
<td>push( )</td>
<td>push( )</td>
</tr>
<tr>
<td>pop( )</td>
<td>pop( )</td>
</tr>
<tr>
<td>peek( )</td>
<td>peek( )</td>
</tr>
<tr>
<td>isEmpty( )</td>
<td>isEmpty( )</td>
</tr>
<tr>
<td>size( )</td>
<td>size( )</td>
</tr>
<tr>
<td>toString( )</td>
<td>toString( )</td>
</tr>
</tbody>
</table>

stack top
UML Diagram for Postfix Expression Program

ArrayStack
- stack
- top
- push()
- pop()
- peek()
- isEmpty()
- size()
- toString()

StackADT
- push()
- pop()
- peek()
- isEmpty()
- size()
- toString()

PostfixEvaluator
- stack
- evaluate()
- isOperator()
- evalSingleOp()

Postfix
- main()
Features of Set of UML Diagrams

• **Inheritance:**
  • An arrow on an association line indicates that one class is derived from the other
Example: Inheritance Relationships

```
LibraryItem
- title
- callNumber
- checkout()
- return()

Book
- author
- publisher

Video
- producer
- studio
```
Example: Inheritance Relationships

BankAccount
- accountNumber
- balance
- deposit()
- withdraw() etc.

SavingsAccount
- interestRate
- addInterest()
- getInterestRate()
- setInterestRate()

CheckingAccount
- transactionCount
- deposit()
- withdraw()
- deductFees()
- getTransactionCount()