KATHOLIEKE UNIVERSITEIT LEUVEN

Department of Computer Science

Associations with Unrestricted Multiplicity

- Data structures
- Representation invariants

Assignment

KATHOLIEKE UNIVERSITEIT LEUVEN

- Design a small system for registering information concerning the purchase of shares
  - A purchase of some share starts with an order involving the number of items to be bought and a highest price
    - As long as an order has not been executed, the order can be cancelled
    - Once an order is executed, the highest price cannot be changed any more
    - Once an order has been executed, it is possible to sell some or all of the items
  - The system will register the current price of each share, together with a 4-letter code identifying its company (e.g. AAPL for Apple)
    - Different shares are assumed to have different 4-letter codes
  - A wallet is a collection of purchases
    - Both purchases that have been ordered and purchases that have been executed are part of a wallet
    - For reasons of simplicity, a wallet cannot have different purchases involving the same share

Assignment

KATHOLIEKE UNIVERSITEIT LEUVEN

- The system must offer at least support for the following functional requirements
  - Put an order for a number of items of a given share at a given highest price
  - Register the execution of an order
  - Sell a given number of items of a given purchase
  - Check whether a given wallet includes at least one item of a given share
    - This function must return its result in (nearly) constant time

Task 1

Overview

KATHOLIEKE UNIVERSITEIT LEUVEN

- Packages for grouping classes
  - In Java, classes that have some things in common can be grouped into packages
- External view on associations with unrestricted connectivity
  - Associations with unrestricted multiplicity can be presented as sets or as lists
- Representation invariants
  - Impose restrictions on internal data structures
- Epilogue

Packages: Goals

KATHOLIEKE UNIVERSITEIT LEUVEN

- In Java, packages are used to group classes with common semantics
  - Examples are packages grouping classes defining banking products, graphical user interfaces, networking, …
    - Names of packages typically do not start with a capital letter
  - Packages can be structured in a hierarchical way with no limits on the depth of nesting
    - In addition to classes, packages can contain (sub)packages
- Packages introduce separate name spaces
  - Within a package, different elements must have different names
    - A class and a package can however have the same name, even if they belong to the same package
  - Elements belonging to different packages can have the same name

Packages: Class path

KATHOLIEKE UNIVERSITEIT LEUVEN

- Packages correspond to directories, in the same way definitions of classes correspond to files
  - An environment variable (“CLASSPATH”) is used (behind the scenes) to locate top-level packages
    - The Java compiler and other tools consult this environment variable
- In professional software, classes always belong to a specific package
  - The default package is only used for purposes of demonstration

Package Statement

KATHOLIEKE UNIVERSITEIT LEUVEN

- The package statement serves to identify the package to which a class belongs
  - The package statement precedes the definition of the class(es) to which it applies
- Packages lead to the notion of fully qualified names
  - The fully qualified name of a class or a package is its own name preceded by the fully qualified name of the package it belongs to
    - The fully qualified name of the predefined class “ArrayList” is “java.util.ArrayList”

Import Statement

KATHOLIEKE UNIVERSITEIT LEUVEN

- Import statements avoid using fully qualified names for classes (and packages)
  - An import statement may specify the fully qualified name of a class
    - An import statement implies that the stated class can be used in an unqualified way throughout the definition of the classes that follow
  - An import statement may also specify that all classes residing in a given package are imported all at once
    - The name of the package is then followed by a “*”
  - Import statements precede the definition of the class(es) to which they apply
- Since Java 1.5, import statements can also be used to avoid qualified names of static methods and variables
  - A static import implies that the stated static element (or all static elements in case of a *) can be used in an unqualified way

Access Rights

KATHOLIEKE UNIVERSITEIT LEUVEN

- The default access right in Java expresses that all classes residing in the same package have access
  - Class ingredients with default access right are referred to as package accessible ingredients
- Classes can also be qualified public or package accessible
  - Ingredients of package accessible classes are only accessible in classes residing in the same package
    - This is even so for public ingredients of such classes

Task 2+3+4

Overview

KATHOLIEKE UNIVERSITEIT LEUVEN

- Packages for grouping classes
  - In Java, classes that have some things in common can be grouped into packages
- External view on associations with unrestricted connectivity
  - Associations with unrestricted multiplicity can be presented as sets or as lists
- Representation invariants
  - Impose restrictions on internal data structures
- Epilogue

List Presentation

KATHOLIEKE UNIVERSITEIT LEUVEN

- Methods
  - Introduce basic inspectors `getXAt(i)` and `getNbXs()`, if associated objects are ordered by position
    - Complement with methods such as `addAsXAt(x,i)` and `removeAsXAt(i)`
      - Methods such as `addAsX(x)` and `removeAsX(x)` are valuable alternatives
    - Number elements in a sequence starting from 1
  - Introduce checkers `canHaveAsXAt(x,i)` or `isValidAsXAt(X,i)`
    - Encapsulate class invariants in `hasProperXs()`
- Example
  - `getGranteeAt(i)`, `getNbGrantees()`, `addAsGranteeAt(person,i)`, `addAsGrantee(person)`, `removeAsGranteeAt(i)`, `removeAsGrantee(person)`, `canHaveAsGranteeAt(person,i)`, `hasProperGrantees()`

Generic Classes

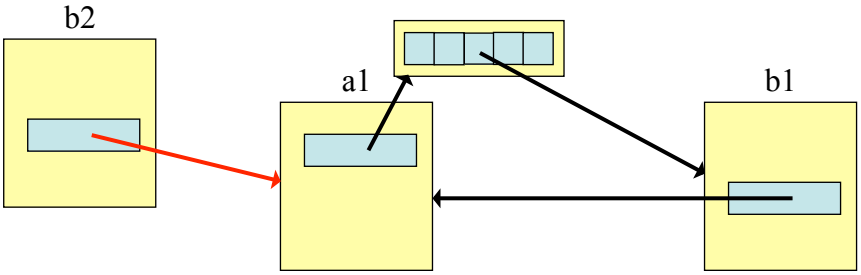
KATHOLIEKE UNIVERSITEIT LEUVEN

- A generic class "ClassName<E_1, E_2, …, E_n>" is parameterized in the types E_1, E_2, …, E_n
  - In the definition of the generic class any of the formal arguments E_1, E_2, …, E_n can be used (almost) anywhere a type can be used
- A generic class is instantiated by supplying specific types for each of its formal arguments
  - An instantiated generic class "ClassName< T_1, T_2, …, T_n>" can be used anywhere an ordinary class can be used
- All objects of all instantiations of a generic class belong to a single class, referred to as the raw type
  - The Java Virtual Machine operates on raw types (classes), and has no knowledge of generic classes
- Examples of generic classes in the Java API are "ArrayList<E>", "List<E>" and "Comparable<T>"

Class Invariants

KATHOLIEKE UNIVERSITEIT LEUVEN

- Consistency must be imposed at both ends of a bi-directional association involving different classes
  - If an object `a1` at the A-side of an association refers to an object `b1` at the B-side, then `b1` must refer back to `a1`
    - This does not prevent an object `b2` from referring the object `a1`, which is not referring back to `b2`
    - A similar invariant is therefore needed at the B-side of the association



Task 5

Set Presentation

KATHOLIEKE UNIVERSITEIT LEUVEN

- Methods
  - Introduce a basic inspector `hasAsX(x)`, if associated objects are unordered
    - Complement with setters `addAsX(x)` and `removeAsX(x)`
  - Introduce checkers `canHaveAsX(x)` or `isValidX(x)`
    - Encapsulate class invariants in `hasProperXs()`
- Example
  - `hasAsSavingsAccount(savings)`
  - `addAsSavingsAccount(savings)`
  - `removeAsSavingsAccount(savings)`
  - `canHaveAsSavingsAccount(savings)`
  - `hasProperSavingsAccounts()`

Task 6

Overview

KATHOLIEKE UNIVERSITEIT LEUVEN

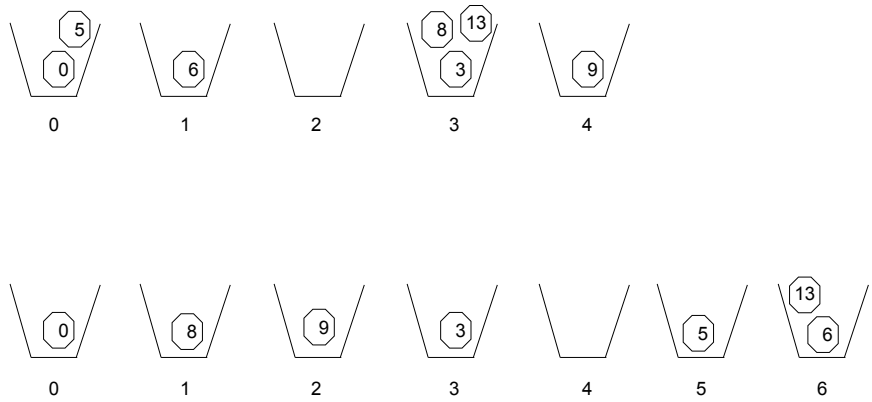
- Packages for grouping classes
  - In Java, classes that have some things in common can be grouped into packages
- External view on associations with unrestricted connectivity
  - Associations with unrestricted multiplicity can be presented as sets or as lists
- Representation invariants
  - Impose restrictions on internal data structures
- Epilogue

Hash Structures

KATHOLIEKE UNIVERSITEIT LEUVEN

- A hash structure involves a number of buckets collecting elements with corresponding hash code
  - The hash code of an object is an integer value that must be specific enough to distinguish it from most other objects
    - The root class “Object” introduces a default definition of the method “hashCode()”
- Elements with identical hash code modulo the number of buckets are stored in the same bucket
  - The hash code of an element determines the bucket in which it must be stored
    - A linear search is used to search for elements in the same bucket
  - As soon as a hash structure is overloaded, the number of buckets is extended (“rehash”)
- The Java API offers classes “HashSet<E>” and “HashMap<K,V>”

Hash Structures



Representation Invariants

- The internal representation of collection objects may differ significantly from their external presentation
  - Internally an ordered collection may be used; externally, the collection may seem to be unordered (or vice versa)
    - For ordered collections, inspections are more efficient in time, while mutators are more time consuming
  - Internally, null-references may be stored as elements of the collection, while externally the collection only has effective objects
    - For collections with null references, mutators (especially removals) are simple; some inspections may be more difficult
  - …

Representation Invariants

KATHOLIEKE UNIVERSITEIT LEUVEN

- Representation invariants impose restrictions on instance variables and class variables
  - Representation invariants must hold upon entry to any non-private instance method or class method
    - Representation invariants must hold again upon exit from each such method
  - Representation invariants are of no interest to clients of a class
- In this course, representation invariants are specified both formally (first-order logic) and informally (natural language)
  - The specification of a representation invariant is worked out as part of the declaration of the variable to which it applies
    - Each representation invariant starts with the non-standard tag "@invar"
    - Vertical bars ("|") separate formal specifications from informal ones

Task 7

Overview

KATHOLIEKE UNIVERSITEIT LEUVEN

- Packages for grouping classes
  - In Java, classes that have some things in common can be grouped into packages
- External view on associations with unrestricted connectivity
  - Associations with unrestricted multiplicity can be presented as sets or as lists
- Internal view on associations with unrestricted connectivity
  - For associations with unrestricted multiplicity, generic data structures are used to collect all associated objects
- Epilogue

Summary

KATHOLIEKE UNIVERSITEIT LEUVEN

- Packages serve to group classes related to the same domain
  - The default package only serves for demonstration purposes
- Collections are used in representing associations with unrestricted connectivity
  - The representation for an association with unrestricted connectivity can differ from its external presentation
- Representation invariants impose restrictions on elements in the internal collection
  - Representation invariants may differ from class invariants
- Constructors, mutators and destructors must manipulate references in both directions
  - Only raw objects may have inconsistent bindings upon exit of a method

Homework

KATHOLIEKE UNIVERSITEIT LEUVEN

- None so far