Check06C

Number of students enrolled in the course: 209

Number of students that eChecked Check05C: 17 (8%)

Check07C

Number of students enrolled in the course: 198

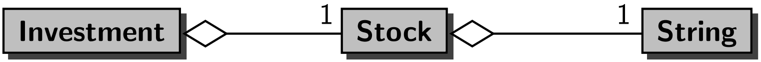
Number of students that eChecked Check07C: 18 (9%)

Check08A

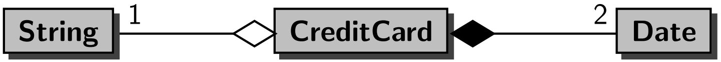
Number of students enrolled in the course: 191

Number of students that eChecked Check08A: 14 (7%)

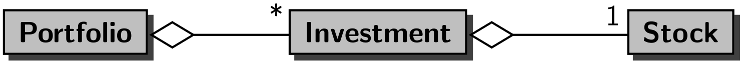
Aggregation



Composition



Collection



Collection

We distinguish between

- static allocation: the maximum number of elements (capacity) is fixed when the collection is created
- dynamic allocation: the number of elements is unbounded

and

- list: duplicates are allowed and the elements are ordered
- set: duplicates are disallowed and the elements are *not* ordered

Traversals

```
for each element of the collection
    ...
```

We distinguish two types of traversals:

- indexed traversals
- Iterator-based traversals

Indexed Traversals

```
... collection = ...
...
for (int i = 0; i < collection.size(); i++)
{
   ... element = collection.get(i);
   ...
}
```

Indexed Traversals

```
... collection = ...
...
for (int i = 0; i < collection.size(); i++)
{
   ... element = collection.get(i);
   ...
}
```

Question

Print all investments (one per line) of a random portfolio.

Indexed Traversals

Question

Which collections support indexed traversals?

Indexed Traversals

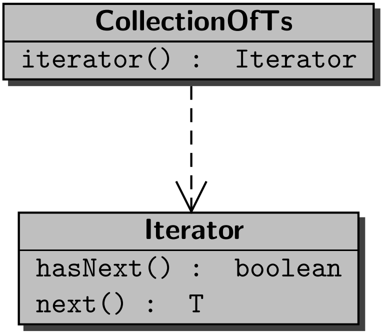
Question

Which collections support indexed traversals?

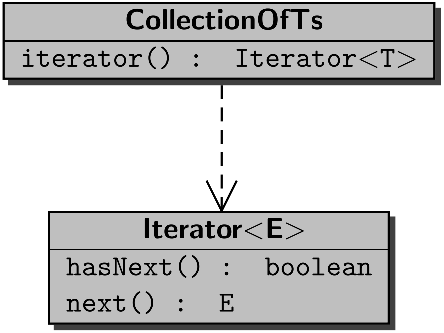
Answer

Those that contain the methods `size()` and `get(int)`.

Iterator-Based Traversals



Iterator-Based Traversals



T is a type (class name) and E is a type parameter (we come back to this in Section 9.3.3).

Iterator-Based Traversals

```
... collection = ...
...
Iterator<...> iterator = collection.iterator();
while (iterator.hasNext())
{
    ... element = iterator.next();
    ...
}
```

The above can be abbreviated using the advanced for loop:

```
... collection = ...
...
for (... element : collection)
{
    ...
}
```

Iterator-Based Traversals

```
... collection = ...
...
for (... element : collection)
{
   ...
}
```

Iterator-Based Traversals

```
... collection = ...
...
for (... element : collection)
{
    ...
}
```

Question

Print all creditcards (one per line) of a random creditcard centre.

Iterator-Based Traversals

Question

Which collections support iterator-based traversals?

Iterator-Based Traversals

Question

Which collections support iterator-based traversals?

Answer

Those that contain the method `iterator()`.

Iterator-Based Traversals

Question

Which collections support iterator-based traversals?

Answer

Those that contain the method `iterator()`.

Question

Which collections support the advanced for loop?

Iterator-Based Traversals

Question

Which collections support iterator-based traversals?

Answer

Those that contain the method `iterator()`.

Question

Which collections support the advanced for loop?

Answer

Those that implement `Iterable` (we come back to this in Chapter 10).

Smallest Integer

Question

Write an app that takes an number of integers are command line arguments and prints the smallest.

Smallest Integer

Question

Which of the two solutions is better?

Smallest Integer

Question

Which of the two solutions is better?

Answer

The app `Fast`.

Smallest Integer

Question

Which of the two solutions is better?

Answer

The app `Fast`.

Question

Why?

Smallest Integer

Question

Which of the two solutions is better?

Answer

The app `Fast`.

Question

Why?

Answer

It is more efficient (that is, it is faster).

Smallest Integer

Question

How would you prove that?

Smallest Integer

Question

How would you prove that?

Answer

Time the app.

Smallest Integer

Question

How would you prove that?

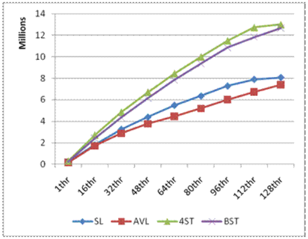
Answer

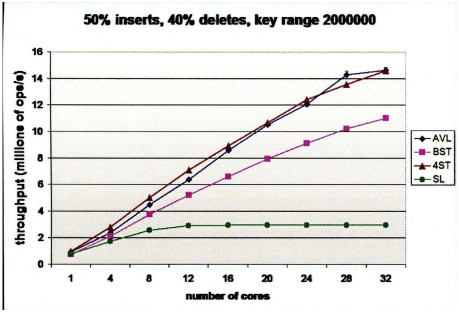
Time the app.

Question

How often? Which inputs?

Which Machine?





Research by undergraduate student Trevor Brown and graduate student Joanna Helga.

Inheritance

Add some simple data to already existing complex data.

For the resulting data, add new operations (mainly to handle the added simple data) and possibly redefine some of the operations of the complex data.

Inheritance

Inheritance was invented in 1967 for the object-oriented programming language Simula.

"Inheritance is an object-oriented technique that allows you to re-use code across related objects in your applications."
Source: www.objectorientedcoldfusion.org/wiki/Inheritance

"Item 14: Favor composition (aggregation) over inheritance."
Source: Joshua Bloch. *Effective Java: Programming Language Guide*. Addison-Wesley. 2001.

Inheritance

Definition

Inheritance is a binary relation on classes. The pair (C, P) of classes is in the inheritance relation if the API of the class C (child) contains

```
class C extends P
```

The API of the class P (parent) may (but does not have to) contain

```
Direct Known Subclasses: C
```

The inheritance relation is also known as the is-a relation. Instead of saying that (C, P) is in the inheritance relation, we often simply say that C is-a P.

Inheritance

Example

`RewardCard` is-a `CreditCard`

`CEStudent` is-a `Student`

`ITStudent` is-a `Student`

`SEStudent` is-a `Student`

Inheritance

Definition

C is a superclass of P if C is-a P.

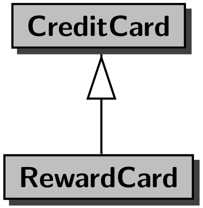
P is a subclass of C if C is-a P.

Example

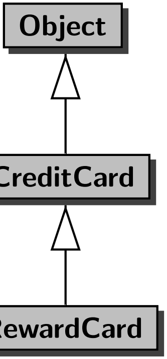
`Student` is a superclass of `CEStudent`

`RewardCard` is a subclass of `CreditCard`

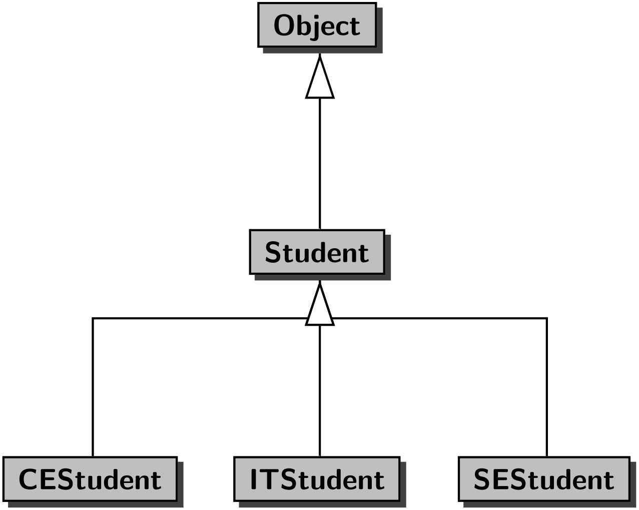
UML Diagrams



UML Diagrams



UML Diagrams



Inheritance

Definition

A programming language supports *single inheritance* if each class has at most one superclass.

A programming language supports *multiple inheritance* if each class may have multiple superclasses.

Example

Java supports single inheritance.

Eiffel supports multiple inheritance.

A Class Consists of

- constructors,
- attributes, and
- methods.

Constructors

Constructors are not inherited from the superclass.

In a Well-Designed Class

- all non-final attributes are private (page 77).

We will restrict ourselves to such well-designed classes.

Attributes

All public non-static final attributes are inherited from the superclass.

For a well-designed class, these are the only non-static attributes of which a client is aware. The other non-static attributes are relevant to the implementer and are considered in CSE1030.

Static attributes are not inherited. They can be accessed via the superclass (name).

Attributes

Question

Assume that the class P has a public non-static final attribute named a. Can its subclass C also contain a public non-static final attribute named a?

Attributes

Question

Assume that the class P has a public non-static final attribute named a. Can its subclass C also contain a public non-static final attribute named a?

Answer

Yes. In that case, the attribute a of the subclass C is said to *shadow* the attribute a of the superclass P.

However, why would one ever introduce two different constants with the same name? In well-designed classes, such a situation never arises.

Methods

All public non-static methods are inherited from the superclass.

The private non-static methods are relevant to the implementer and are considered in CSE1030.

Static methods are not inherited. They can be invoked via the superclass (name).

Methods

Question

Assume that the class P has a public non-static method with signature (method name and parameter types) s. Can its subclass C also contain a public non-static method with signature s?

Methods

Question

Assume that the class P has a public non-static method with signature (method name and parameter types) s. Can its subclass C also contain a public non-static method with signature s?

Answer

Yes. In that case, the method with signature s of the subclass C is said to *override* the method with signature s of the superclass P.

Methods

We can distinguish between

- inherited methods
- overridden methods
- new methods

Drop deadline

November 12

Until this date you can drop the course without getting a grade for it and, hence, it will not affect your gpa.