

Programación Orientada a Objetos

Colecciones

CEIS

Noviembre 2017

Agenda

- 1 Introducción
 - Conceptos
 - Ejemplos
- 2 Oferta java
 - Manejo
 - Representación
 - Selección
- 3 Operaciones java
 - Básicas
 - Analizadoras
 - Ejemplos
 - De soporte
- 4 Colecciones propias
 - Alternativas
 - Ejemplos
 - Colecciones genericas
- 5 Variedad

Agenda

1 Introducción

Conceptos

Ejemplos

2 Oferta java

Manejo

Representación

Selección

3 Operaciones java

Básicas

Analizadoras

Ejemplos

De soporte

4 Colecciones propias

Alternativas

Ejemplos

Colecciones genericas

5 Variedad

Conceptos

En general

Un colección es un tipo especial de datos usado para almacenar y organizar otros datos

¿Qué colecciones conocen?

- PIMB - PIMO
- LCAL - MDIS

¿Cómo las categorizamos?

Conceptos

Operaciones-básicas

Operaciones-analizadoras

Conceptos

Operaciones-básicas

- Crear
- Adicionar un elemento a la colección
- Eliminar un elemento de la colección

Operaciones-analizadoras

Operaciones-básicas

- Crear
- Adicionar un elemento a la colección
- Eliminar un elemento de la colección

Operaciones-analizadoras

- ¿Cuántos elementos hay en la colección?
- ¿Qué elemento está en una posición de la colección?

Introducción

Conceptos

Ejemplos

Oferta java

Manejo

Representación

Selección

**Operaciones
java**

Básicas

Analizadoras

Ejemplos

De soporte

**Colecciones
propias**

Alternativas

Ejemplos

Colecciones

genericas

Variedad

Operaciones-básicas

- Crear
- Adicionar un elemento a la colección
- Eliminar un elemento de la colección

Operaciones-analizadoras

- ¿Cuántos elementos hay en la colección?
- ¿Qué elemento está en una posición de la colección?
-
- ¿Está un elemento en la colección?

Conceptos

En POOB

Un colección es un tipo especial de objetos usado para almacenar y organizar referencias a otros objetos

¿Qué colecciones hemos manejado?

Agenda

1 Introducción

Conceptos

Ejemplos

2 Oferta java

Manejo

Representación

Selección

3 Operaciones java

Básicas

Analizadoras

Ejemplos

De soporte

4 Colecciones propias

Alternativas

Ejemplos

Colecciones genericas

5 Variedad

Laboratorios

Calculadora

```
public class Calculadora{  
    private Stack<Vector> operandos;  
    //Consultar en el API Java la clase Stack
```

BodyTic

```
public class Salon{  
    private ArrayList<EnSalon> elementos;
```

Galeria

```
public class GaleriaTIC{  
    private LinkedList <Personaje> personajes;
```

¿Qué contienen?

¿Qué permiten?

Agenda

1 Introducción

Conceptos

Ejemplos

2 Oferta java

Manejo

Representación

Selección

3 Operaciones java

Básicas

Analizadoras

Ejemplos

De soporte

4 Colecciones propias

Alternativas

Ejemplos

Colecciones genericas

5 Variedad

Conceptos

Tipos básicos

- 1 Colecciones simples
- 2 Colecciones con clave

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

Tipos básicos

- 1 Colecciones simples
 - Pueden existir elementos repetidos, interesa la posición.
 - No pueden existir elemento repetidos
- 2 Colecciones con clave

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

Tipos básicos

- ① Colecciones simples
 - Pueden existir elementos repetidos, interesa la posición.
 - No pueden existir elemento repetidos
- ② Colecciones con clave
 - No pueden existir claves repetidas

Dos objetos base

Contexto java

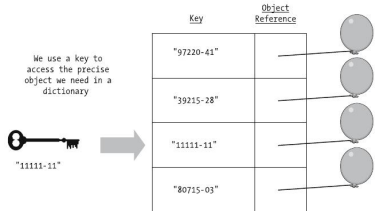
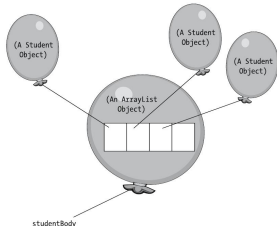
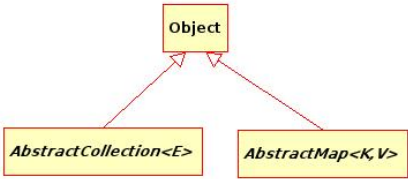
Introducción
Conceptos
Ejemplos

Oferta java
Manejo
Representación
Selección

Operaciones
java
Básicas
Analizadoras
Ejemplos
De soporte

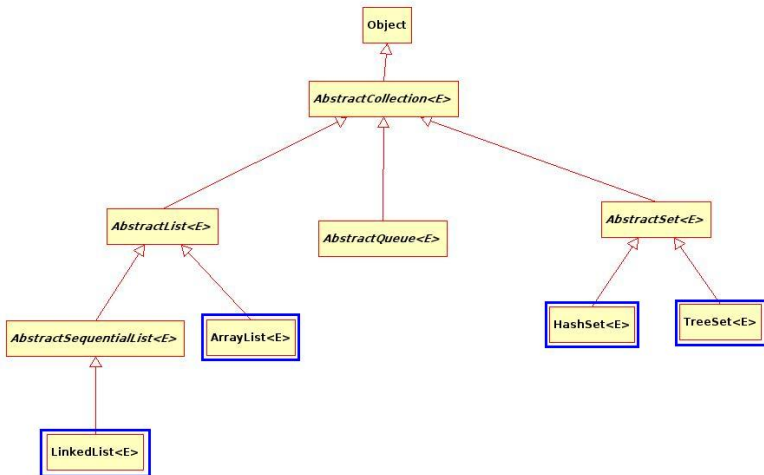
Colecciones
propias
Alternativas
Ejemplos
Colecciones
genericas

Variedad



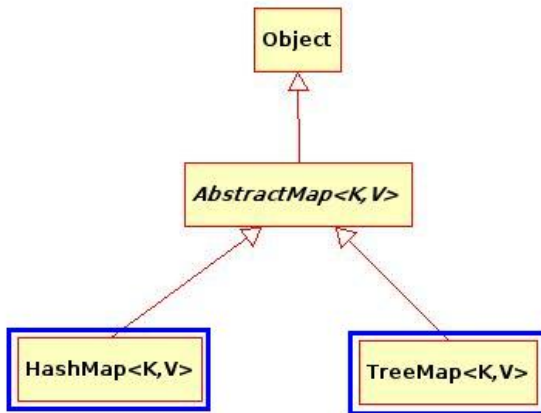
Contexto java

Colecciones



Contexto java

Mapeos



Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

Tipos básicos

- 1 Colecciones simples
 - Pueden existir elementos repetidos.
 - No pueden existir elemento repetidos
- 2 Colecciones con clave

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

Tipos básicos

① Colecciones simples

- Pueden existir elementos repetidos.
List: **Array****List**, **Linked****List**
- No pueden existir elemento repetidos

② Colecciones con clave

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones
java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones
propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

Tipos básicos

① Colecciones simples

- Pueden existir elementos repetidos.

List: Array**List**, Linked**List**

- No pueden existir elemento repetidos

Set: Hash**Set**, Tree**Set**

② Colecciones con clave

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones
java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones
propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

Tipos básicos

① Colecciones simples

- Pueden existir elementos repetidos.
List: **ArrayList**, **LinkedList**
- No pueden existir elemento repetidos
Set: **HashSet**, **TreeSet**

② Colecciones con clave

Map: **HashMap**, **TreeMap**

Agenda

1 Introducción

Conceptos

Ejemplos

2 Oferta java

Manejo

Representación

Selección

3 Operaciones java

Básicas

Analizadoras

Ejemplos

De soporte

4 Colecciones propias

Alternativas

Ejemplos

Colecciones genericas

5 Variedad

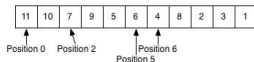
Representación

Diferenciadores

- 1 **Array**
ArrayList
- 2 **Linked**
LinkedList
- 3 **Hash**
HashSet, **Hash**Map
- 4 **Tree**
TreeSet, **Tree**Map

Representación

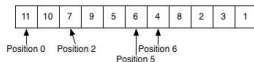
Array



Arreglo que puede cambiar de tamaño.

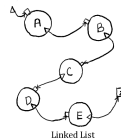
Representación

Array



Arreglo que puede cambiar de tamaño.

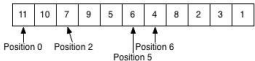
List



Lista de apuntadores.

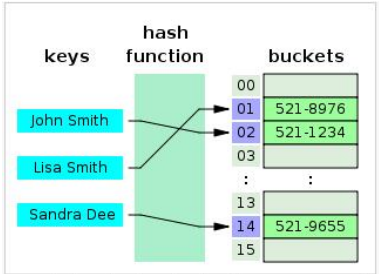
Representación

Array



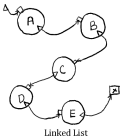
Arreglo que puede cambiar de tamaño.

Hash



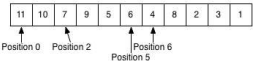
A small phone book as a hash table.

List



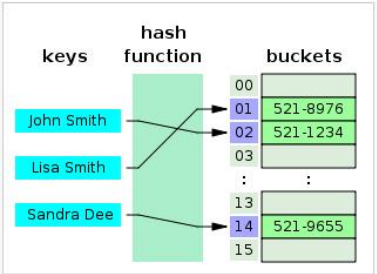
Lista de apuntadores.

Array



Arreglo que puede cambiar de tamaño.

Hash

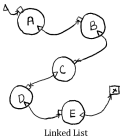


A small phone book as a hash table.



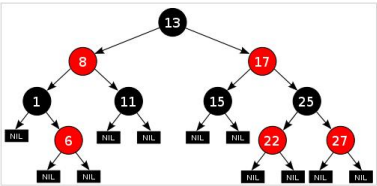
Representación

List



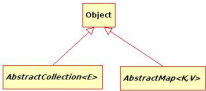
Lista de apuntadores.

Tree

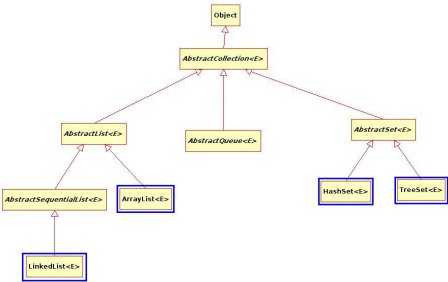


Contexto java

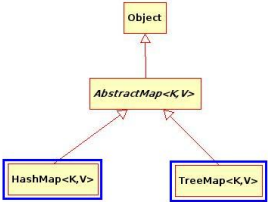
Dos objetos base



Sin clave



Con clave



Agenda

1 Introducción

Conceptos

Ejemplos

2 Oferta java

Manejo

Representación

Selección

3 Operaciones java

Básicas

Analizadoras

Ejemplos

De soporte

4 Colecciones propias

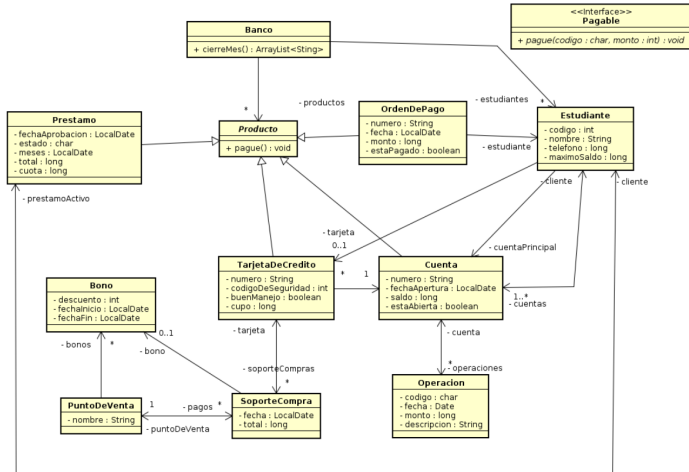
Alternativas

Ejemplos

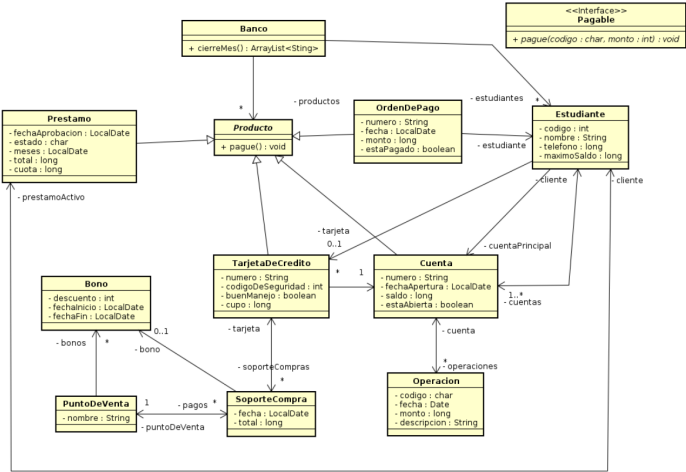
Colecciones genericas

5 Variedad

Parcial



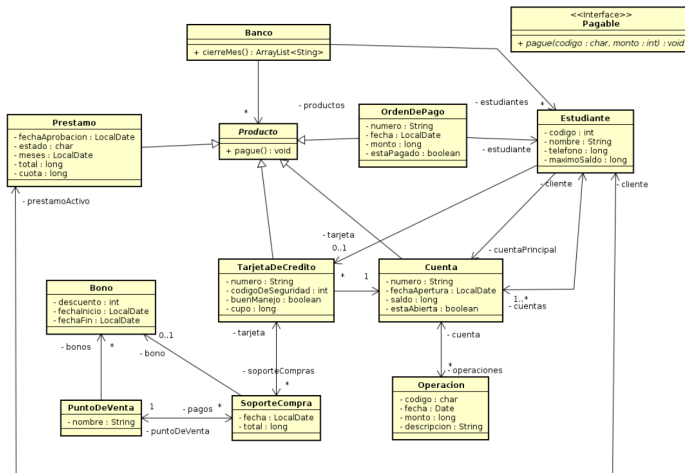
Diseñando



Diseñando

Los soporte de compras los queremos consultar por fecha Los estudiantes las queremos consultar por código y por nombre. Además, los queremos ordenados por nombres

Parcial



Diseñando

Los productos los queremos consultar por tipo.

Agenda

1 Introducción

Conceptos

Ejemplos

2 Oferta java

Manejo

Representación

Selección

3 Operaciones java

Básicas

Analizadoras

Ejemplos

De soporte

4 Colecciones propias

Alternativas

Ejemplos

Colecciones genericas

5 Variedad

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

Operaciones-básicas

- **Crear**
- Adicionar un objeto a la colección
- Eliminar un objeto de la colección

Operaciones-analizadoras

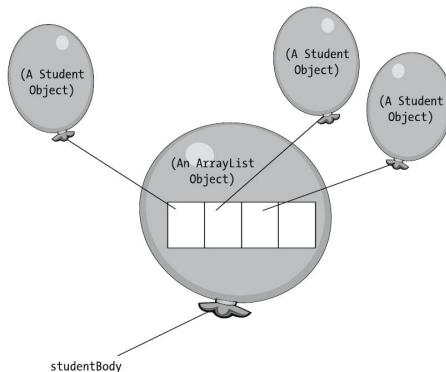
- ¿Cuántos elementos hay en la colección?
- ¿Qué elemento está en una posición de la colección?
- ¿Está un elemento específico en la colección?

En general

Creando

```
ArrayList<Student> studentBody ; // ArrayList is one of Java's predefined collec  
studentBody = new ArrayList<Student>();
```

En Uso



Operaciones básicas

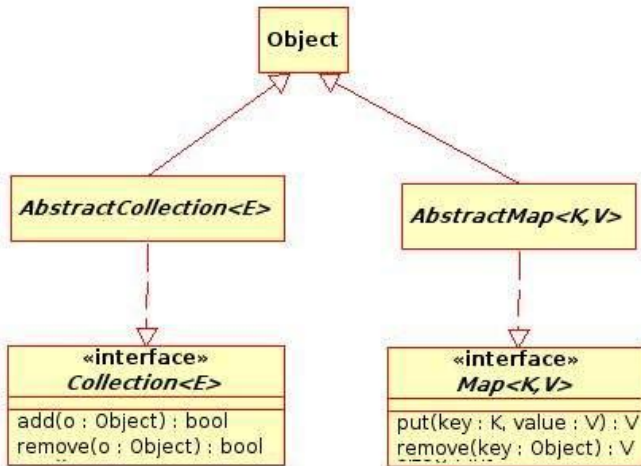
- Crear
- **Adicionar un objeto a la colección**
- **Eliminar un objeto de la colección**

Operaciones-analizadoras

- ¿Cuántos elementos hay en la colección?
- ¿Qué elemento está en una posición de la colección?
- ¿Está un elemento en la colección?

Contexto java

Operaciones básicas



Agenda

- 1 **Introducción**
 - Conceptos
 - Ejemplos
- 2 **Oferta java**
 - Manejo
 - Representación
 - Selección
- 3 **Operaciones java**
 - Básicas
 - Analizadoras**
 - Ejemplos
 - De soporte
- 4 **Colecciones propias**
 - Alternativas
 - Ejemplos
 - Colecciones genericas
- 5 **Variedad**

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

Operaciones básicas

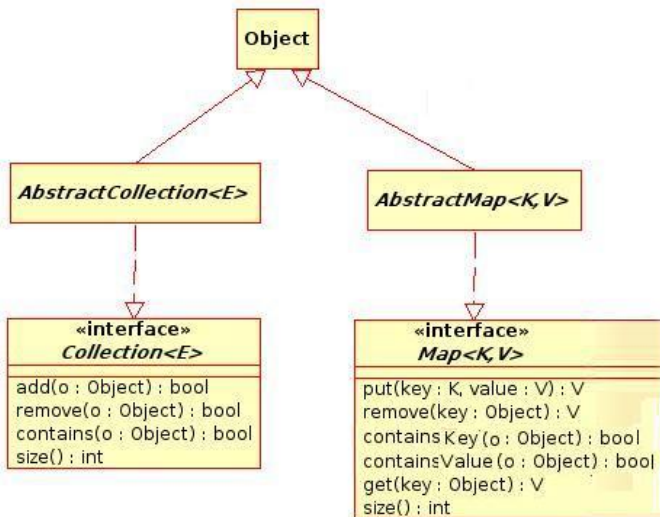
- Crear
- Adicionar un objeto a la colección
- Eliminar un objeto de la colección

Operaciones-analizadoras

- ¿Cuántos elementos hay en la colección?
- ¿Qué elemento está en una posición de la colección?
- ¿Está un elemento en la colección?

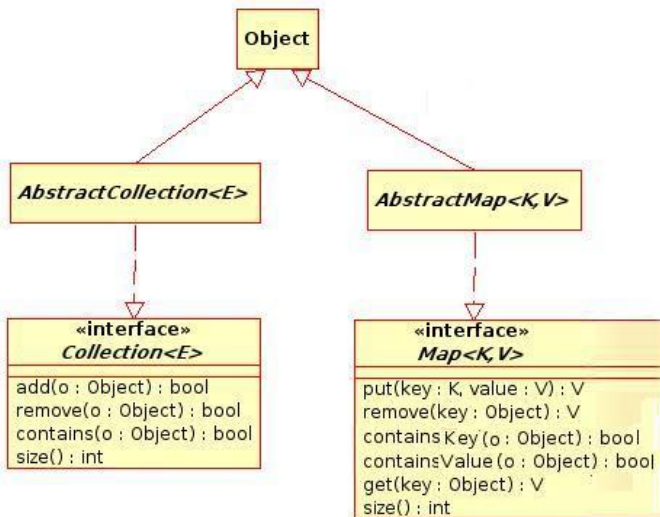
Contexto java

Operaciones básicas



Contexto java

Operaciones básicas



Conceptos

Operaciones básicas

- Crear
- Adicionar un objeto a la colección
- Eliminar un objeto de la colección

Operaciones-analizadoras

- ¿Cuántos elementos hay en la colección?
- ¿Qué elemento está en una posición de la colección?
- ¿Está un elemento específico en la colección?
-
- ¿Cuáles son cada uno de los objetos que hay en la colección?

Contexto java

Recorriendo

```
for (type referenceVariable : collectionName) {  
    // Pseudocode.  
    manipulate the referenceVariable as desired  
}
```

Especificación

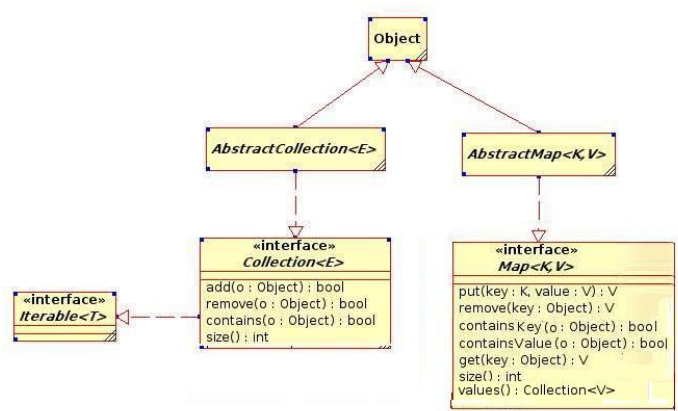
The enhanced for statement has the form:

```
EnhancedForStatement:  
    for ( VariableModifiersopt Type Identifier: Expression) Statement
```

The *Expression* must either have type `Iterable` or else it must be of an array type ([§10.1](#)), or a compile-time error occurs.

Recorriendo

- Introducción
 - Conceptos
 - Ejemplos
- Oferta java
 - Manejo
 - Representación
 - Selección
- Operaciones java
 - Básicas
- Analizadoras
 - Ejemplos
 - De soporte
- Colecciones propias
 - Alternativas
 - Ejemplos
 - Colecciones genericas
- Variedad



Iterable

Interface Iterable<T>

Method Summary

<u>Iterator</u> <T>	<u>iterator</u> () Returns an iterator over a set of elements of type T.
---------------------	---

Method Detail

iterator

Iterator<T> **iterator**()
Returns an iterator over a set of elements of type T.

Returns:
an Iterator.

Iterable

java.util

Interface Iterator<E>

Method Summary

boolean	hasNext()	Returns true if the iteration has more elements.
E	next()	Returns the next element in the iteration.

Conceptos

Operaciones básicas

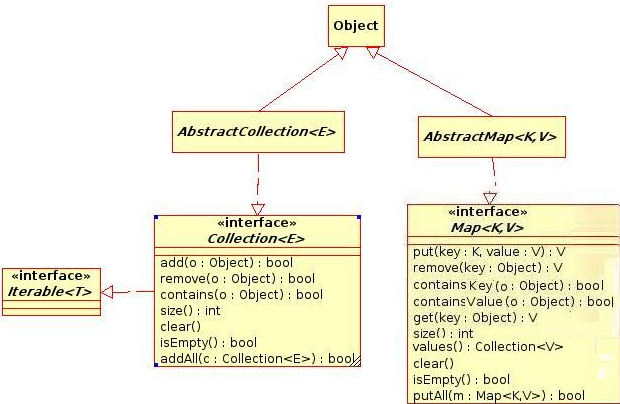
- Crear
- Adicionar un objeto a la colección
- Eliminar un objeto de la colección

Operaciones-analizadoras

- ¿Cuántos elementos hay en la colección?
- ¿Qué elemento está en una posición de la colección?
-
- ¿Está un elemento específico en la colección?

Otras operaciones

Otras operaciones



Agenda

1 Introducción

Conceptos

Ejemplos

2 Oferta java

Manejo

Representación

Selección

3 Operaciones java

Básicas

Analizadoras

Ejemplos

De soporte

4 Colecciones propias

Alternativas

Ejemplos

Colecciones genericas

5 Variedad

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras

Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

```
import java.util.*;

public class ArrayListExample {
    public static void main(String[] args) {
        // Instantiate a collection.
        ArrayList<Student> students = new ArrayList<Student>();

        // Create a few Student objects.
        Student a = new Student();
        Student b = new Student();
        Student c = new Student();

        // Store references to all three Students in the collection.
        students.add(a);
        students.add(b);
        students.add(c);

        // ... and then iterate through them one by one,
        // printing each student's name.
        for (Student s : students) {
            System.out.println(s.getName());
        }
    }
}
```

En general

```
import java.util.*;

public class ArrayListExample {
    public static void main(String[] args) {
        // Instantiate a collection.
        ArrayList<Student> students = new ArrayList<Student>();

        // Create a few Student objects.
        Student a = new Student();
        Student b = new Student();
        Student c = new Student();

        // Store references to all three Students in the collection.
        students.add(a);
        students.add(b);
        students.add(c);

        // ... and then iterate through them one by one,
        // printing each student's name.
        for (Student s : students) {
            System.out.println(s.getName());
        }
    }
}
```

¿De qué otro tipo puede ser students sin cambiar código?

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras

Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

```
import java.util.HashMap;

public class HashMapExample {
    public static void main(String[] args) {
        // Instantiate a HashMap with String as the key type and Student as
        // the value type.
        HashMap<String, Student> students = new HashMap<String, Student>();

        // Instantiate three Students; the constructor arguments are
        // used to initialize Student attributes idNo and name,
        // respectively, which are both declared to be Strings.
        Student s1 = new Student("12345-12", "Fred");
        Student s2 = new Student("98765-00", "Barney");
        Student s3 = new Student("71024-91", "Wilma");

        // Insert all three Students into the HashMap, using their idNo
        // as a key.
        students.put(s1.getIdNo(), s1);
        students.put(s2.getIdNo(), s2);
        students.put(s3.getIdNo(), s3);
    }
}
```

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras

Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

```
// Retrieve a Student based on a particular (valid) ID.
String id = "98765-00";
System.out.println("Let's try to retrieve a Student with ID = " + id);
Student x = students.get(id);
if (x != null) {
    System.out.println("Found! Name = " + x.getName());
}
// ... whereas if the value returned was null, then we didn't find
// a match on the id that was passed in as an argument to get().
else {
    System.out.println("Invalid ID: " + id);
}
```

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras

Ejemplos
De soporte

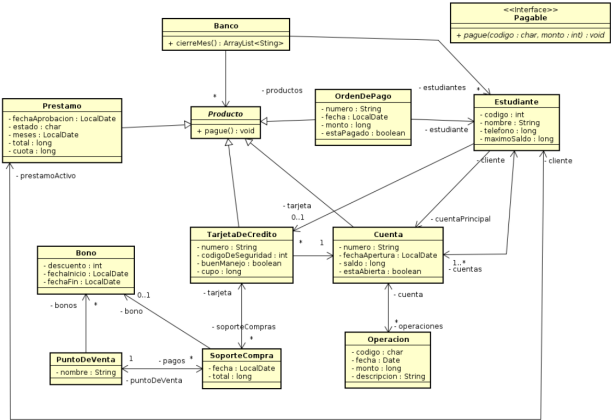
Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

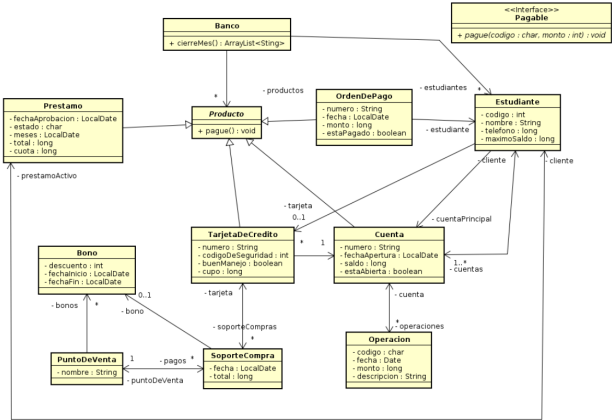
Variedad

```
System.out.println();  
System.out.println("Here are all of the students:");  
System.out.println();  
  
// Iterate through the HashMap to process all Students.  
for (Student s : students.values()) {  
    System.out.println("ID: " + s.getIdNo());  
    System.out.println("Name: " + s.getName());  
    System.out.println();  
}
```

¿De qué otro tipo puede ser Students sin cambiar código?

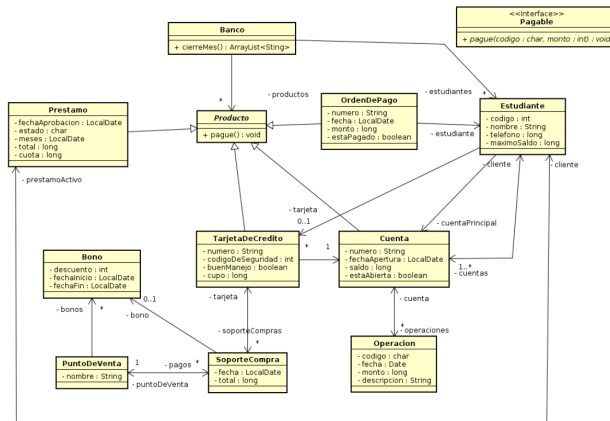


Usando



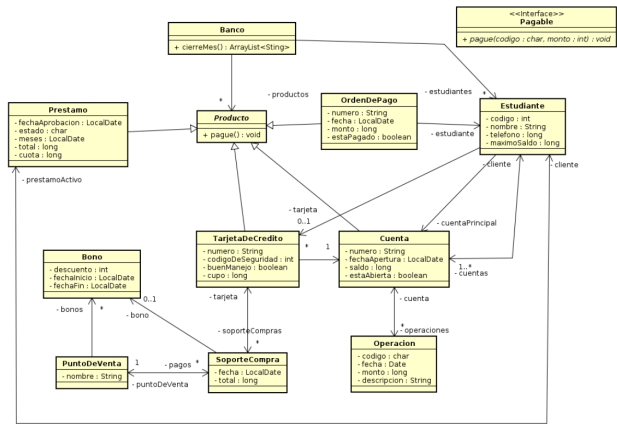
Usando

Adicionar un soporte de compra xSoporte



Usando

¿Todos los estudiantes con saldos máximos mayor del promedio en orden alfabético estudiante?



Usando

¿Número de productos por tipos?.

Agenda

- 1 Introducción
 - Conceptos
 - Ejemplos
- 2 Oferta java
 - Manejo
 - Representación
 - Selección
- 3 Operaciones java
 - Básicas
 - Analizadoras
 - Ejemplos
 - De soporte**
- 4 Colecciones propias
 - Alternativas
 - Ejemplos
 - Colecciones genericas
- 5 Variedad

Object

Constructor Summary

[`Object\(\)`](#)

Method Summary

boolean	<code>equals(Object obj)</code> Indicates whether some other object is "equal to" this one.
int	<code>hashCode()</code> Returns a hash code value for the object.

Object

Constructor Summary

[Object\(\)](#)

Method Summary

boolean	equals(Object obj) Indicates whether some other object is "equal to" this one.
int	hashCode() Returns a hash code value for the object.

Todos usan **equals**. Si es necesario se debe definir.

Object

Constructor Summary

[Object\(\)](#)

Method Summary

boolean	equals(Object obj) Indicates whether some other object is "equal to" this one.
int	hashCode() Returns a hash code value for the object.

Todos usan **equals**. Si es necesario se debe definir.
Las Hash usan **hashCode**

Comparable

java.lang

Interface Comparable<T>

Method Summary

int	compareTo (T o)	Compares this object with the specified object for order.
-----	------------------------	---

Method Detail

compareTo

int **compareTo**(T o)

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

¿2-¿ Los elementos del Tree deben implementar la interfaz Comparable

Agenda

- 1 Introducción
 - Conceptos
 - Ejemplos
- 2 Oferta java
 - Manejo
 - Representación
 - Selección
- 3 Operaciones java
 - Básicas
 - Analizadoras
 - Ejemplos
 - De soporte
- 4 Colecciones propias
 - Alternativas**
 - Ejemplos
 - Colecciones genericas
- 5 Variedad

Colecciones propias

Aproximaciones

- 1 Crear la clase desde cero
- 2 Extender una clase colección predefinida
- 3 Crear una clase que tenga como un atributo la colección predefinida

Colecciones propias

Opción dos

```
import java.util.ArrayList;

public class MyIntCollection extends ArrayList<Integer> {
    private int smallestInt;
    private int largestInt;
    private int total;

    public MyIntCollection() {
        super();
        total = 0;
    }

    public boolean add(int i) {
        if (this.isEmpty()) {
            smallestInt = i;
            largestInt = i;
        }
        else {
            if (i < smallestInt) smallestInt = i;
            if (i > largestInt) largestInt = i;
        }

        total = total + i;
        return super.add(i);
    }
}
```

Colecciones propias

Opción dos

```
// Several new methods.

public int getSmallestInt() {
    return smallestInt;
}

public int getLargestInt() {
    return largestInt;
}

public double getAverage() {
    // Note that we must cast ints to doubles to avoid
    // truncation when dividing.
    return ((double) total) / ((double) this.size());
}
}
```

Colecciones propias

Opción dos

```
// Several new methods.

public int getSmallestInt() {
    return smallestInt;
}

public int getLargestInt() {
    return largestInt;
}

public double getAverage() {
    // Note that we must cast ints to doubles to avoid
    // truncation when dividing.
    return ((double) total) / ((double) this.size());
}
```

¿Qué es bueno? ¿Qué es malo?

Opción tres

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

```
import java.util.ArrayList;
public class MyIntCollection2 {
    // Instead, we're encapsulating a ArrayList inside of this class.
    ArrayList<Integer> numbers;
    private int smallestInt;
    private int largestInt;
    private int total;
    public MyIntCollection2() {
        numbers = new ArrayList<Integer>();
        total = 0;
    }
    public boolean add(int i) {
        if (numbers.isEmpty()) {
            smallestInt = i;
            largestInt = i;
        }
        else {
            if (i < smallestInt) smallestInt = i;
            if (i > largestInt) largestInt = i;
        }
        total = total + i;
        return numbers.add(i);
    }
}
```

Colecciones propias

Opción tres

```
public int getSmallestInt() {  
    return smallestInt;  
}  
  
public int getLargestInt() {  
    return largestInt;  
}  
  
public double getAverage() {  
    return ((double) total)/this.size();  
}  
  
// Since we don't INHERIT a size() method any longer, let's add one!  
public int size() {  
    // DELEGATION!  
    return numbers.size();  
}  
}
```

Colecciones propias

Opción tres

```
public int getSmallestInt() {  
    return smallestInt;  
}  
  
public int getLargestInt() {  
    return largestInt;  
}  
  
public double getAverage() {  
    return ((double) total)/this.size();  
}  
  
// Since we don't INHERIT a size() method any longer, let's add one!  
public int size() {  
    // DELEGATION!  
    return numbers.size();  
}  
}
```

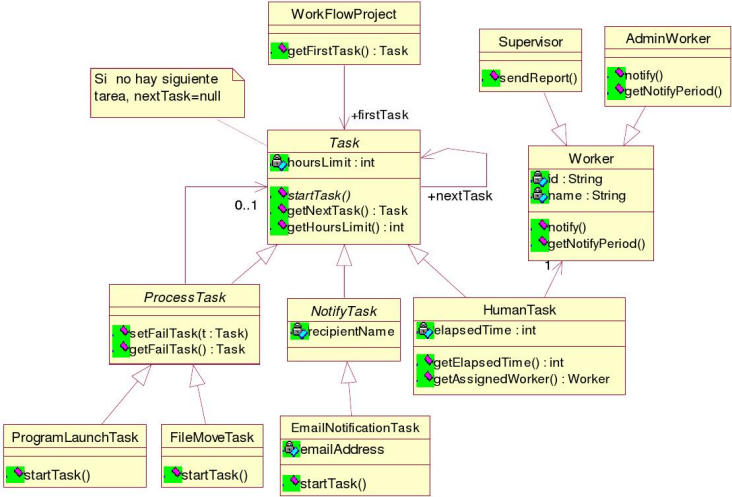
¿Qué es bueno? ¿Qué es malo?

Agenda

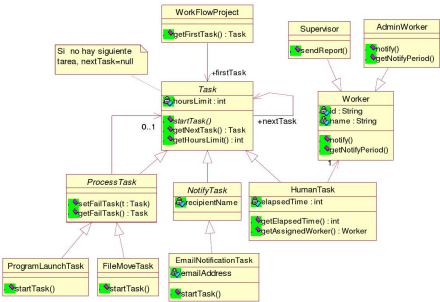
- 1 **Introducción**
 - Conceptos
 - Ejemplos
- 2 **Oferta java**
 - Manejo
 - Representación
 - Selección
- 3 **Operaciones java**
 - Básicas
 - Analizadoras
 - Ejemplos
 - De soporte
- 4 **Colecciones propias**
 - Alternativas
 - Ejemplos**
 - Colecciones genericas
- 5 **Variedad**

Flujo de trabajo

Ejemplos



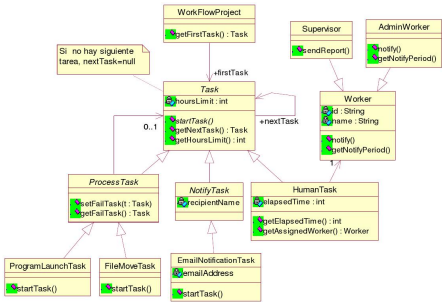
Flujo de trabajo



Analizando

1 ¿Qué colección tenemos?

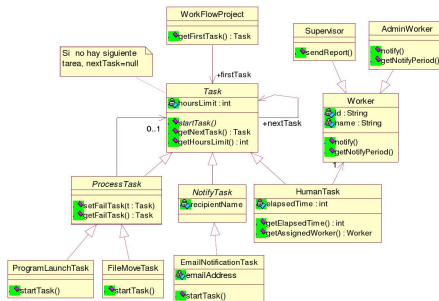
Flujo de trabajo



Analizando

- 1 ¿Qué colección tenemos?
- 2 ¿Qué alternativa se seleccionó?

Flujo de trabajo



Analizando

- ❶ ¿Qué colección tenemos?
- ❷ ¿Qué alternativa se seleccionó?
- ❸ ¿Cómo lo haríamos considerando otra alternativa?

Agenda

- 1 Introducción
 - Conceptos
 - Ejemplos
- 2 Oferta java
 - Manejo
 - Representación
 - Selección
- 3 Operaciones java
 - Básicas
 - Analizadoras
 - Ejemplos
 - De soporte
- 4 Colecciones propias
 - Alternativas
 - Ejemplos
 - Colecciones genericas
- 5 Variedad

Colecciones propias

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

```
import java.util.ArrayList;
public class MyIntCollection2 {
    // Instead, we're encapsulating a ArrayList inside of this class.
    ArrayList<Integer> numbers;
    private int smallestInt;
    private int largestInt;
    private int total;
    public MyIntCollection2() {
        numbers = new ArrayList<Integer>();
        total = 0;
    }
    public boolean add(int i) {
        if (numbers.isEmpty()) {
            smallestInt = i;
            largestInt = i;
        }
        else {
            if (i < smallestInt) smallestInt = i;
            if (i > largestInt) largestInt = i;
        }
        total = total + i;
        return numbers.add(i);
    }
}
```

Colecciones propias

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

```
import java.util.ArrayList;
public class MyIntCollection2 {
    // Instead, we're encapsulating a ArrayList inside of this class.
    ArrayList<Integer> numbers;
    private int smallestInt;
    private int largestInt;
    private int total;
    public MyIntCollection2() {
        numbers = new ArrayList<Integer>();
        total = 0;
    }
    public boolean add(int i) {
        if (numbers.isEmpty()) {
            smallestInt = i;
            largestInt = i;
        }
        else {
            if (i < smallestInt) smallestInt = i;
            if (i > largestInt) largestInt = i;
        }
        total = total + i;
        return numbers.add(i);
    }
}
```


Colecciones propias

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

```
import java.util.ArrayList;
public class MyIntCollection2 {

    ArrayList<Integer> numbers;
    private int smallestInt;
    private int largestInt;

    public MyIntCollection2() {
        numbers = new ArrayList<Integer>();
    }

    public boolean add(int i) {
        if (numbers.isEmpty()) {
            smallestInt = i;
            largestInt = i;
        }
        else {
            if (i < smallestInt) smallestInt = i;
            if (i > largestInt) largestInt = i;
        }

        return numbers.add(i);
    }
}
```

Generica

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

```
import java.util.ArrayList;

public class MyCollection <E> {
    private ArrayList <E> collection;
    private E largest;
    private E smallest;

    public MyCollection () {
        collection=new ArrayList <E>();
        largest=null;
        smallest=null;
    }

    public boolean add (E element){
        if (collection.isEmpty()){
            largest=element;
            smallest=element;
        } else {
            if (element.compareTo(largest)>0) largest=element;
            if (element.compareTo(smallest) <0) smallest=element;
        }
        return collection.add(element);
    }
}
```

Generica

Introducción

Conceptos
Ejemplos

Oferta java

Manejo
Representación
Selección

Operaciones java

Básicas
Analizadoras
Ejemplos
De soporte

Colecciones propias

Alternativas
Ejemplos
Colecciones
genericas

Variedad

```
import java.util.ArrayList;

public class MyCollection <E> {
    private ArrayList <E> collection;
    private E largest;
    private E smallest;

    public MyCollection () {
        collection=new ArrayList <E>();
        largest=null;
        smallest=null;
    }

    public boolean add (E element){
        if (collection.isEmpty()){
            largest=element;
            smallest=element;
        } else {
            if (element.compareTo(largest)>0) largest=element;
            if (element.compareTo(smallest) <0) smallest=element;
        }
        return collection.add(element);
    }
}
```

Generica

```
import java.util.ArrayList;
import java.lang.Comparable;

public class MyCollection <E extends Comparable<E>>{
    private ArrayList <E> collection;
    private E largest;
    private E smallest;

    public MyCollection (){
        collection=new ArrayList <E>();
        largest=null;
        smallest=null;
    }

    public boolean add (E element){
        if ((collection.isEmpty()) || (element.compareTo(largest)>0)){
            largest=element;
        }
        if ((collection.isEmpty()) || (element.compareTo(largest)<0)){
            smallest=element;
        }
        return collection.add(element);
    }
}
```

Isla Coco

Requisitos

- 1 Necesitamos almacenar la información de los tesoros $(x,y,\$)$
- 2 Existen muy pocos tesoros en la isla
- 3 Los queremos consultar por posición y ordenados por filas $((1,1)..(1,n),...(m,1),(m,n))$

Isla Coco

Requisitos

- 1 Necesitamos almacenar la información de los tesoros $(x,y,\$)$
- 2 Existen muy pocos tesoros en la isla
- 3 Los queremos consultar por posición y ordenados por filas $((1,1)..(1,n),...(m,1),(m,n))$

Isla Coco

Requisitos

- 1 Necesitamos almacenar la información de los tesoros $(x,y,\$)$
- 2 Existen muy pocos tesoros en la isla
- 3 Los queremos consultar por posición y ordenados por filas $((1,1)..(1,n),... (m,1),(m,n))$
- 4 ¿Y si lo queremos ordenado también por columnas?

Otro orden

Constructor Summary	
<code>TreeMap()</code>	Constructs a new, empty map, sorted according to the keys' natural order.
<code>TreeMap(Comparator<? super K> c)</code>	Constructs a new, empty map, sorted according to the given comparator.
<code>TreeMap(Map<? extends K, ? extends V> m)</code>	Constructs a new map containing the same mappings as the given map, sorted according to the keys' <i>natural order</i> .
<code>TreeMap(SortedMap<K, ? extends V> m)</code>	Constructs a new map containing the same mappings as the given <code>SortedMap</code> , sorted according to the same ordering.

Otro orden

Comparator

Method Summary	
int	<code>compare</code> (<code>T</code> o1, <code>T</code> o2) Compares its two arguments for order.
boolean	<code>equals</code> (<code>Object</code> obj) Indicates whether some other object is "equal to" this Comparator.