Bộ môn Công nghệ Phần mềm
Viện CNTT & TT
Trường Đại học Bách Khoa Hà Nội

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
Bài 07. Đa hình (Polymophism)

Nội dung

1. Upcasting và Downcasting
2. Liên kết tĩnh và Liên kết động
3. Đa hình (Polymophism)
4. Lập trình tổng quát (generic prog.)

Nội dung

1. Upcasting và Downcasting
2. Liên kết tĩnh và Liên kết động
3. Đa hình (Polymophism)
4. Lập trình tổng quát (generic prog.)

1.1. Upcasting

- Moving up the inheritance hierarchy

Ví dụ

```
public class Test1 {
  public static void main(String arg[]){
     Person p;
     Employee e = new Employee();
     p = e;
     p.setName("Hoa");
     p.setSalary(350000);
}
```


Ví dụ (2)

```
class Manager extends Employee {
  Employee assistant;
  // ...
  public void setAssistant(Employee e) {
      assistant = e;
  }
  // ...
}
public class Test2 {
  public static void main(String arg[]){
      Manager junior, senior;
      // ...
      senior.setAssistant(junior);
  }
}
```

Ví dụ (3)

```
public class Test3 {
  String static teamInfo(Person p1, Person p2){
      return "Leader: " + p1.getName() +
             ", member: " + p2.getName();
  }

  public static void main(String arg[]){
      Employee e1, e2;
      Manager m1, m2;
      // ...
      System.out.println(teamInfo(e1, e2));
      System.out.println(teamInfo(m1, m2));
      System.out.println(teamInfo(m1, e2));
  }
}
```

7

1.2. Downcasting

- Move back down the inheritance hierarchy

8

Ví dụ

```
public class Test2 {
  public static void main(String arg[]){
      Employee e = new Employee();
      Person p = e;
      Employee ee = (Employee) p;
      Manager m = (Manager) ee;
      Person p2 = new Manager();
      Employee e2 = (Employee) p2;

      Person p3 = new Employee();
      Manager e3 = (Manager) p3;
  }
}
```

9

Nội dung

1. Upcasting và Downcasting
2. Liên kết tĩnh và Liên kết động
3. Đa hình (Polymophism)
4. Lập trình tổng quát (generic prog.)

10

2.1. Liên kết tĩnh (Static Binding)

- Liên kết tại thời điểm biên dịch

11

Ví dụ

```
public class Test {
  public static void main(String arg[]){
      Person p = new Person();
      p.setName("Hoa");
      p.setSalary(350000);
  }
}
```

12

2.2. Liên kết động (Dynamic binding)

- Lời gọi phương thức được quyết định khi thực hiện (run-time)

13

Ví dụ

```
public class Test {
  public static void main(String arg[]){
    Person p = new Person();
    // ...
    Employee e = new Employee();
    // ...
    Manager m = new Manager();
    // ...
    Person pArr[] = {p, e, m};
    for (int i=0; i< pArr.length; i++){
      System.out.println(
            pArr[i].getDetail());
    }
  }
}
```

14

Nội dung

1. Upcasting và Downcasting
2. Liên kết tĩnh và Liên kết động
3. Đa hình (Polymophism)
4. Lập trình tổng quát (generic prog.)

15

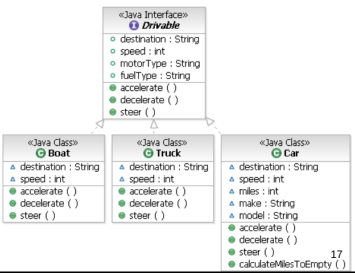
3. Đa hình (Polymophism)

- Ví dụ: Nếu đi du lịch, bạn có thể chọn ô tô, thuyền, hoặc máy bay



16

3. Đa hình (2)



17

3. Đa hình (3)

- Đa hình trong lập trình
  - Đa hình phương thức:
  - Đa hình đối tượng

18

3. Đa hình (4)

```
public class Test3 {
  public static void main(String
                      args[]){
    Person p1 = new Employee();
    Person p2 = new Manager();

    Employee e = (Employee) p1;
    Manager m = (Manager) p2;
  }
}
```

19

3. Đa hình (5)

- Liên kết động
- Ví dụ:

```
Person p1 = new Person();
Person p2 = new Employee();
Person p3 = new Manager();
// ...
System.out.println(p1.getDetail());
System.out.println(p2.getDetail());
System.out.println(p3.getDetail());
```

20

Ví dụ khác

```
class EmployeeList {
  Employee list[];
  ...
  public void add(Employee e) {...}
  public void print() {
    for (int i=0; i<list.length; i++) {

    System.out.println(list[i].getDetail());
    }
}
 ...
  EmployeeList list = new EmployeeList();
  Employee e1; Manager m1;
  ...
  list.add(e1); list.add(m1);
  list.print();
```

21

Toán tử instanceof

```
public class Employee extends Person {}
public class Student extends Person {}

public class Test{
  public doSomething(Person e) {
    if (e instanceof Employee) {...
    } else if (e instanceof Student) {... ){
    } else {...}
  }
}
```

22

Nội dung

1. Upcasting và Downcasting
2. Liên kết tĩnh và Liên kết động
3. Đa hình (Polymophism)
4. Lập trình tổng quát (generic prog.)

23

4. Lập trình tổng quát

- 4.1. Giới thiệu
- 4.2. Java generic data structure
  - 4.2.1. Data structure
  - 4.2.2. Java collection framework
  - 4.2.3. Các interface trong Java collection framework
  - 4.2.4. Các cài đặt cho các interface – implementation
- 4.3. Định nghĩa và sử dụng Template
- 4.4. Ký tự đại diện (Wildcard)

24

4. Lập trình tổng quát

25

4. 1. Giới thiệu về lập trình tổng quát

- C: dùng con trỏ void
- C++: dùng template
- Java: lợi dụng upcasting
- Java 1.5: template

26

Ví dụ: C dùng con trỏ void

```
void* memcpy(void* region1,
           const void* region2, size_t n){
  const char* first = (const char*)region2;
  const char* last = ((const char*)region2) + n;
  char* result = (char*)region1;
  while (first != last)
     *result++ = *first++;
  return result;
}
```

27

Ví dụ: C++ dùng template

```
template<class ItemType>
void sort(ItemType A[], int count ) {
for (int i = count-1; i > 0; i--) {
     int index_of_max = 0;
     for (int j = 1; j <= i ; j++)
        if (A[j] > A[index_of_max]) index_of_max = j;
     if (index_of_max !=  i) {
       ItemType temp = A[i];
       A[i] = A[index_of_max];
       A[index_of_max ] = temp;
     }
  }
}
```

28

Ví dụ: Java dùng upcasting và Object

```
class MyStack {
  ...
  public void push(Object obj) {...}
  public Object pop() {...}
}
public class TestStack{
  MyStack s = new MyStack();
  Point p = new Point();
  Circle c = new Circle();
  s.push(p); s.push(c);
  Circle c1 = (Circle) s.pop();
  Point p1 = (Point) s.pop();
}
```

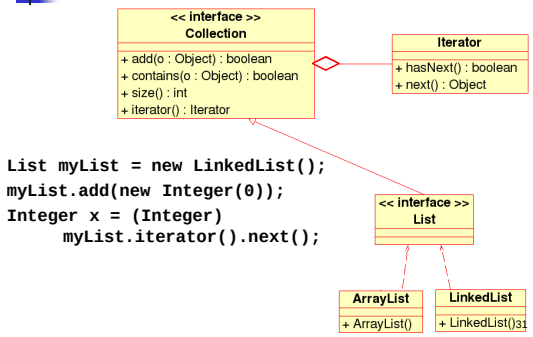
29

Nhắc lại – equals của lớp tự viết

```
class MyValue {
  int i;
}
public class EqualsMethod2 {
  public static void main(String[] args) {
     MyValue v1 = new MyValue();
     MyValue v2 = new MyValue();
     v1.i = v2.i = 100;
     System.out.println(v1.equals(v2));
     System.out.println(v1==v2);
  }
}
```

30

Ví dụ: Java 1.5: Template

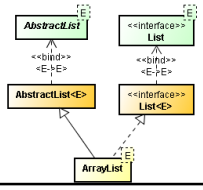



```
List myList = new LinkedList();
myList.add(new Integer(0));
Integer x = (Integer)
      myList.iterator().next();
```

31

Ví dụ: Java 1.5: Template (2)

```
List<Integer> myList = new LinkedList<Integer>();
myList.add(new Integer(0));
Integer x = myList.iterator().next();
```



32

4. Lập trình tổng quát

- 4.1. Giới thiệu
- 4.2. Java generic data structure
  - 4.2.1. Data structure
  - 4.2.2. Java collection framework
  - 4.2.3. Các interface trong Java collection framework
  - 4.2.4. Các cài đặt cho các interface – implementation
- 4.3. Định nghĩa và sử dụng Template
- 4.4. Ký tự đại diện (Wildcard)

33

4.2.1. Cấu trúc dữ liệu-data structure

- Mảng (Array)
- Danh sách liên kết (Linked List)
- Ngăn xếp (Stack)
- Hàng đợi (Queue)
- Cây (Tree)

34

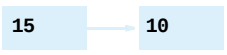
a. Linked List

- Khi chèn/xoá một node trên linked list, không phải dãn/dồn các phần tử như trên mảng.

35

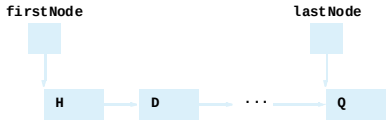
a. Linked List (2)

```
class Node
{
   private int data;
   private Node nextNode;
   // constructors and methods ...
}
```



36

a. Linked List (3)



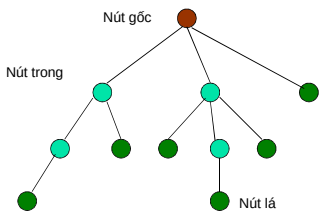
37

b. Stack

- Stack là một cấu trúc theo kiểu LIFO (Last In First Out), phần tử vào sau cùng sẽ được lấy ra trước.

38

c. Tree



39

d. Queue

- Queue (Hàng đợi) là cấu trúc theo kiểu FIFO

40

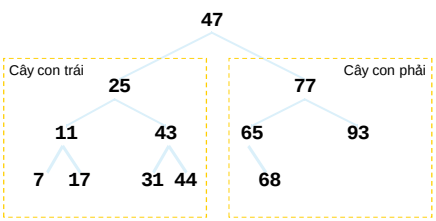
e. Binary Search Tree

- Cây nhị phân là cây mà mỗi node không có quá 2 node con.
- Cây tìm kiếm nhị phân

41

e. Binary Search Tree (2)

- Ví dụ về Binary Search Tree



42

4. Lập trình tổng quát

- 4.1. Giới thiệu
- 4.2. Java generic data structure
  - 4.2.1. Data structure
  - 4.2.2. Java collection framework
  - 4.2.3. Các interface trong Java collection framework
  - 4.2.4. Các cài đặt cho các interface – implementation
- 4.3. Định nghĩa và sử dụng Template
- 4.4. Ký tự đại diện (Wildcard)

43

4.2.2. Java Collection Framework

- Collection là đối tượng có khả năng chứa các đối tượng khác.

44

4.2.2. Java Collection Framework (2)

- Các collection đầu tiên của Java:
- Collections Framework (từ Java 1.2)

45

4.2.2. Java Collection Framework (3)

- Một số lợi ích của Collections Framework

46

4.2.2. Java Collection Framework (4)

- Collections Framework bao gồm
  - Interfaces:
  - Implementations:
  - Algorithms:

47

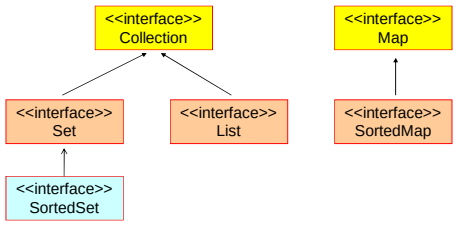
4. Lập trình tổng quát

- 4.1. Giới thiệu
- 4.2. Java generic data structure
  - 4.2.1. Data structure
  - 4.2.2. Java collection framework
  - 4.2.3. Các interface trong Java collection framework
  - 4.2.4. Các cài đặt cho các interface – implementation
- 4.3. Định nghĩa và sử dụng Template
- 4.4. Ký tự đại diện (Wildcard)

48

4.2.3. Interfaces

- List:
- Set:
- Map:



49

a. Giao diện Collection

50

b. Giao diện List

- Một số phương thức của List
  - Object get(int index);
  - Object set(int index, Object o);
  - void add(int index, Object o);
  - Object remove(int index);
  - int indexOf(Object o);
  - int lastIndexOf(Object o);

51

c. Giao diện Set

- Set kế thừa từ Collection

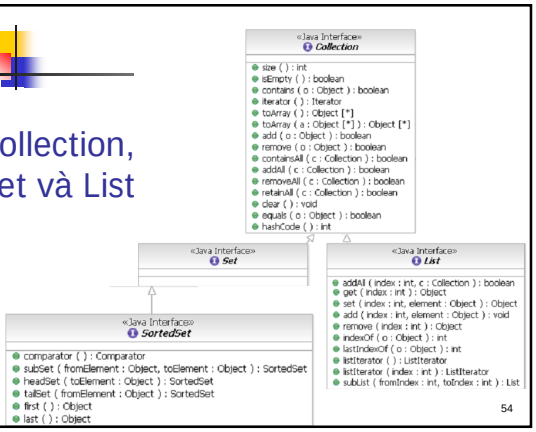
52

d. Giao diện SortedSet

- SortedSet kế thừa từ Set
- Một số phương thức của SortedSet:
  - Object first();
  - Object last
  - SortedSet subSet(Object e1, Object e2);

53

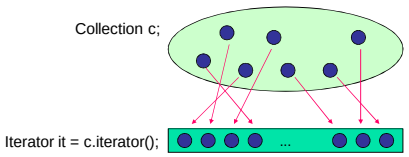
Collection, Set và List



54

e. Duyệt collection

- Iterator



55

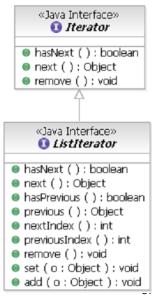
e. Duyệt collection (2)

- Các phương thức của Iterator:
  - boolean hasNext();
  - Object next();
  - void remove();

```
Iterator it = c.iterator();
while ( it.hasNext() ) {
   Point p = (Point) it.next();
   System.out.println( p.toString() );
}
```

56

f. Giao diện Iterator



f. Giao diện Iterator (2) - Ví dụ

```
Collection c;
// Some code to build the
  collection

Iterator i = c.iterator();
while (i.hasNext()) {
  Object o = i.next();
  // Process this object
}
```

58

g. Giao diện Map

- Xác định giao diện cơ bản để thao tác với một tập hợp bao gồm cặp khóa-giá trị

59

g. Giao tiếp Map (2)

- Map cung cấp 3 cách view dữ liệu

60

h. Giao diện SortedMap

- Giao diện SortedMap kế thừa từ Map, nó cung cấp thao tác trên các bảng ánh xạ với khoá có thể so sánh được.

61

4. Lập trình tổng quát

- 4.1. Giới thiệu
- 4.2. Java generic data structure
  - 4.2.1. Data structure
  - 4.2.2. Java collection framework
  - 4.2.3. Các interface trong Java collection framework
  - 4.2.4. Các cài đặt cho các interface – implementation
- 4.3. Định nghĩa và sử dụng Template
- 4.4. Ký tự đại diện (Wildcard)

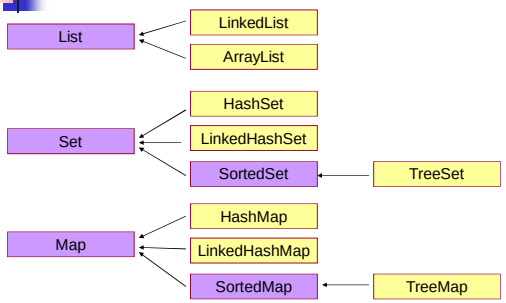
62

4.2.4. Implementations

- Các cài đặt trong Collections Framework chính là các lớp collection có sẵn trong Java.

63

4.2.4. Implementations (2)



64

4.2.4. Implementations (3) -Mô tả các cài đặt

- ArrayList:
- LinkedList
- HashSet:
- LinkedHashSet:
- TreeSet:

65

4.2.4. Implementations (3) -Mô tả các cài đặt

- HashMap:
- LinkedHashMap:
- TreeMap:

66

4.2.4. Implementations (3) – Tổng kết

		IMPLEMENTATIONS				
		Hash Table	Resizable Array	Balanced Tree	Linked List	Legacy
INTERFACES	Set	HashSet		TreeSet		
	List		ArrayList		LinkedList	Vector, Stack
	Map	HashMap		TreeMap		HashTable, Properties

```
public class MapExample {
  public static void main(String args[]) {
      Map map = new HashMap();
      Integer ONE = new Integer(1);
      for (int i=0, n=args.length; i<n; i++) {
        String key = args[i];
        Integer frequency =(Integer)map.get(key);
        if (frequency == null) { frequency = ONE; }
        else {
          int value = frequency.intValue();
          frequency = new Integer(value + 1);
        }
        map.put(key, frequency);
      }
      System.out.println(map);
      Map sortedMap = new TreeMap(map);
      System.out.println(sortedMap);
  }
}
```

68

4. Lập trình tổng quát

- 4.1. Giới thiệu
- 4.2. Java generic data structure
  - 4.2.1. Data structure
  - 4.2.2. Java collection framework
  - 4.2.3. Các interface trong Java collection framework
  - 4.2.4. Các cài đặt cho các interface – implementation
- 4.3. Định nghĩa và sử dụng Template
- 4.4. Ký tự đại diện (Wildcard)

69

4.3. Định nghĩa và sử dụng Template

```
class MyStack<T> {
    ...
    public void push(T x) {...}
    public T pop() {
        ...
    }
}
```

70

Sử dụng template

```
public class Test {
  public static void main(String args[]) {

      MyStack<Integer> s1 = new MyStack<Integer>();
      s1.push(new Integer(0));
      Integer x = s1.pop();

      //s1.push(new Long(0)); → Error

      MyStack<Long> s2 = new MyStack<Long>();
      s2.push(new Long(0));
      Long y = s2.pop();

  }
}
```

71

Định nghĩa Iterator

```
public interface List<E>{
    void add(E x);
    Iterator<E> iterator();
}

public interface Iterator<E>{
    E next();
    boolean hasNext();
}
class LinkedList<E> implements List<E> {
// implementation
}
```

72

4. Lập trình tổng quát

73

4.4. Ký tự đại diện (Wildcard)

```
public class Test {
  public static void main(String args[]) {
     List<String> lst0 = new LinkedList<String>();
     //List<Object> lst1 = lst0; → Error
     //printList(lst0); → Error
  }

  void printList(List<Object> lst) {
     Iterator it = lst.iterator();
     while (it.hasNext())
          System.out.println(it.next());
  }
}
```

74

Ví dụ: Sử dụng Wildcards

```
public class Test {
  void printList(List<?> lst) {
     Iterator it = lst.iterator();
     while (it.hasNext())
        System.out.println(it.next());
  }

  public static void main(String args[]) {
     List<String> lst0 =
                new LinkedList<String>();
     List<Employee> lst1 =
                new LinkedList<Employee>();

     printList(lst0);   // String
     printList(lst1);   // Employee
  }
}
```

75

Các ký tự đại diện Java 1.5

- "? extends Type".
- "? super Type"
- "?"

76

Ví dụ wildcard (1)

```
public void printCollection(Collection c) {
   Iterator i = c.iterator();
   for(int k = 0;k<c.size();k++) {
      System.out.println(i.next());
   }
}
```

→ Sử dụng wildcard:

```
void printCollection(Collection<?> c) {
   for(Object o:c) {
      System.out.println(o);
   }
}
```

77

Ví dụ wildcard (2)

```
public void draw(List<Shape> shape) {
   for(Shape s: shape) {
      s.draw(this);
   }
 }
```

78