
HUST
ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
ONE LOVE. ONE FUTURE.

1


ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY
Bài 4: Khởi tạo và sử dụng đối tượng
ONE LOVE. ONE FUTURE.

2

Nội dung

1. Phương thức khởi tạo
2. Các loại phương thức khởi tạo
3. Khai báo và khởi tạo đối tượng
4. Sử dụng đối tượng
5. Quản lý bộ nhớ và so sánh đối tượng
6. Hủy bỏ đối tượng
7. Ví dụ và bài tập

3

Bài giảng E-learning

- Phương thức khởi tạo, khai báo và sử dụng đối tượng
  - https://www.youtube.com/watch?v=rw_bPkesNH0
  - https://www.youtube.com/watch?v=MTCXgdBLrlw
  - https://www.youtube.com/watch?v=XznNdY3Bfvg
- Quản lý bộ nhớ: Stack và Heap
  - https://www.youtube.com/watch?v=450maTzSIvA
  - https://www.youtube.com/watch?v=1rLHJJqx98Q
- Equals và ==
  - https://www.youtube.com/watch?v=qQe69w1YF54
- Java finalize method
  - https://www.youtube.com/watch?v=j3fRK7T1pQo

4

Nội dung

1. **Phương thức khởi tạo**
2. Các loại phương thức khởi tạo
3. Khai báo và khởi tạo đối tượng
4. Sử dụng đối tượng
5. Quản lý bộ nhớ và so sánh đối tượng
6. Hủy bỏ đối tượng
7. Ví dụ và bài tập

5

1. Phương thức khởi tạo

- Dữ liệu cần được khởi tạo trước khi sử dụng
  - Lỗi khởi tạo là một trong các lỗi phổ biến
- Với kiểu dữ liệu đơn giản, sử dụng toán tử =
- Với đối tượng → Cần dùng phương thức khởi tạo

Student
- name
- address
- studentID
- dateOfBirth

6

Khởi tạo và hủy bỏ đối tượng

- Mỗi đối tượng khi tồn tại và hoạt động được hệ điều hành cấp phát một vùng nhớ để lưu lại các giá trị của dữ liệu thành phần
- Khi tạo ra đối tượng HĐH sẽ gán giá trị khởi tạo cho các dữ liệu thành phần
  - Phải được thực hiện tự động trước khi người lập trình có thể tác động lên đối tượng
  - Sử dụng hàm/phương thức khởi tạo
- Ngược lại khi kết thúc cần phải giải phóng hợp lý tất cả các bộ nhớ đã cấp phát cho đối tượng.
  - Java: JVM
  - C++: Hàm hủy (destructor)

1. Phương thức khởi tạo

- Là phương thức đặc biệt được gọi tự động khi tạo ra đối tượng
- Mục đích chính: Khởi tạo cho các thuộc tính của đối tượng

1. Phương thức khởi tạo

- Mỗi lớp phải chứa ít nhất một constructor
  - Có nhiệm vụ tạo ra một thể hiện mới của lớp
  - Tên của constructor trùng với tên của lớp
  - Constructor không có kiểu dữ liệu trả về
- Ví dụ:

```
public BankAccount(String o, double b){
        owner = o;
        balance = b;
}
```


1. Phương thức khởi tạo

- Phương thức khởi tạo **có thể dùng** các chỉ định truy cập
  - **`public`**
  - **`private`**
  - Không có (mặc định – phạm vi package)
- Một phương thức khởi tạo **không thể dùng** các từ khóa **`abstract`**, **`static`**, **`final`**, **`native`**, **`synchronized`**.
- Các phương thức khởi tạo không được xem như là *thành viên của lớp*.

Nội dung

1. Phương thức khởi tạo
2. **Các loại phương thức khởi tạo**
3. Khai báo và khởi tạo đối tượng
4. Sử dụng đối tượng
5. Quản lý bộ nhớ và so sánh đối tượng
6. Hủy bỏ đối tượng
7. Ví dụ và bài tập

2. Các loại phương thức khởi tạo

- 2 loại phương thức khởi tạo
  - Phương thức khởi tạo mặc định (Phương thức khởi tạo không tham số)
  - Phương thức khởi tạo có tham số

Phương khởi tạo mặc định (default constructor)

- Là phương thức khởi tạo **KHÔNG THAM SỐ**

```
public BankAccount(){
     owner = "noname"; balance = 100000;
}
```

- Một lớp nên có phương thức khởi tạo mặc định

Phương thức khởi tạo mặc định

- Khi LTV không viết một phương khởi tạo nào trong lớp
  - JVM cung cấp phương thức khởi tạo mặc định
  - Phương thức khởi tạo mặc định do JVM cung cấp có chỉ định truy cập giống như lớp của nó

```
public class MyClass{
   public static void main(String args){
      //...
   }
}
```

```
public class MyClass{
   public MyClass(){
   }
   public static void main(String args){
      //...
   }
}
```

Phương thức khởi tạo có tham số

- Một phương thức khởi dựng có thể có các tham số truyền vào
- Dùng khi muốn khởi tạo giá trị cho các thuộc tính
- Ví dụ:

```
public BankAccount(String o, double b){
    owner = o;
    balance = b;
}
```


Nội dung

1. Phương thức khởi tạo
2. Các loại phương thức khởi tạo
3. **Khai báo và khởi tạo đối tượng**
4. Sử dụng đối tượng
5. Quản lý bộ nhớ và so sánh đối tượng
6. Hủy bỏ đối tượng
7. Ví dụ và bài tập

3. Khai báo và khởi tạo đối tượng

- Đối tượng được tạo ra, thể hiện hóa (instantiate) từ một mẫu chung (lớp).
- Các đối tượng phải được khai báo *kiểu* của đối tượng trước khi sử dụng:
  - Kiểu của đối tượng là lớp các đối tượng
  - Ví dụ:
    - `String strName;`
    - `BankAccount acc;`

3. Khai báo và khởi tạo đối tượng

- Đối tượng cần được khởi tạo trước khi sử dụng
  - Sử dụng toán tử = để gán
  - Sử dụng từ khóa **`new`** với constructor để khởi tạo đối tượng:
    - Từ khóa **`new`** dùng để tạo ra một đối tượng mới
    - Tự động gọi phương thức khởi tạo tương ứng
  - Một đối tượng được khởi tạo mặc định là **`null`**
- Đối tượng được thao tác thông qua *tham chiếu* (~ con trỏ).
- Ví dụ:

```
BankAccount acc1;
acc1 = new BankAccount();
```

3. Khai báo và khởi tạo đối tượng

- Có thể kết hợp vừa khai báo và khởi tạo đối tượng
- Cú pháp:

```
Ten_lop ten_doi_tuong = new
        Pthuc_khoi_tao(ds_tham_so);
```

- Ví dụ:

```
BankAccount account = new BankAccount();
```


3. Khai báo và khởi tạo đối tượng

- Phương thức khởi tạo **không có giá trị trả về**, nhưng khi sử dụng với từ khóa **new** trả về một tham chiếu đến đối tượng mới

```
public BankAccount(String name) {
   setOwner(name);
}
```

Constructor definition

Constructor use

```
BankAccount account = new BankAccount("Joe Smith");
```

3. Khai báo và khởi tạo đối tượng

- Mảng các đối tượng được khai báo giống như mảng dữ liệu cơ bản
- Mảng các đối tượng được khởi tạo mặc định với giá trị **`null`**.
- Ví dụ:

```
Employee emp1 = new Employee(123456);
Employee emp2;
emp2 = emp1;
Department dept[] = new Department[100];
Test[] t = {new Test(1),new Test(2)};
```


Ví dụ 1

```
class BankAccount{
	private String owner;
	private double balance;
}
public class Test{
	public static void main(String args[]){
		BankAccount acc1 = new BankAccount();
	}
}
```

→ Phương thức khởi tạo mặc định do Java cung cấp.

Ví dụ 2

```
public class BackAccount{
        private String owner;
        private double balance;
        public BankAccount(){
                owner = "noname";
        }
}
public class Test{
        public static void main(String args[]){
                BankAccount acc1 = new BankAccount();
        }
}
```

→ Phương thức khởi tạo mặc định tự viết.

Ví dụ 3

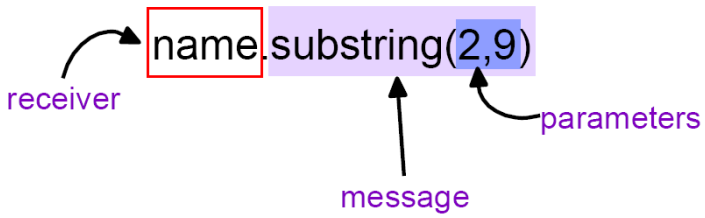
```
public class BankAccount {
   private String owner;
   private double balance;
   public BankAccount(String name) {
      setOwner(name);
   }
   public void setOwner(String o) {
      owner = o;
   }
}
public class Test {
   public static void main(String args[]){
      BankAccount account1 = new BankAccount();     //Error
      BankAccount account2 = new BankAccount("Hoang");
   }
}
```

Nội dung

1. Phương thức khởi tạo
2. Các loại phương thức khởi tạo
3. Khai báo và khởi tạo đối tượng
4. **Sử dụng đối tượng**
5. Quản lý bộ nhớ và so sánh đối tượng
6. Hủy bỏ đối tượng
7. Ví dụ và bài tập

4. Sử dụng đối tượng

- Đối tượng cung cấp các hoạt động phức tạp hơn các kiểu dữ liệu nguyên thủy
- Đối tượng đáp ứng lại các thông điệp
  - Toán tử "." được sử dụng để gửi một thông điệp đến một đối tượng



4. Sử dụng đối tượng (2)

- Để gọi thành viên (dữ liệu hoặc thuộc tính) của lớp hoặc đối tượng, sử dụng toán tử "."
- Nếu gọi phương thức ngay trong lớp thì toán tử "." không cần thiết.

```
BankAccount account = new BankAccount();
account.setOwner("Smith");
account.credit(1000.0);
System.out.println(account.getBalance());
...
```

BankAccount method

```
public void credit(double amount) {
   setBalance(getBalance() + amount);
}
```


Ví dụ

```
public class BankAccount{
   private String owner;
   private double balance;
   public BankAccount(String name) {
     setOwner(name);
     // Là viết tắt của this.setOwner(name)
   }
   public void setOwner(String o){ owner = o; }
   public String getOwner(){ return owner; }
}
public class Test{
   public static void main(String args[]){
    BankAccount acc1 = new BankAccount("");
    BankAccount acc2 = new BankAccount("Hong");
    acc1.setOwner("Hoa");
    System.out.println(acc1.getOwner()
                       + " "+ acc2.getOwner());
}
```

Tự tham chiếu – this

- Cho phép truy cập vào đối tượng hiện tại của lớp.
- Quan trọng khi hàm/phương thức thành phần thao tác trên hai hay nhiều đối tượng.
- Xóa đi sự nhập nhằng giữa một biến cục bộ, tham số với thành phần dữ liệu của lớp
- Không dùng bên trong các khối lệnh static

Ví dụ

```
public class BankAccount{
    private String owner;
    private double balance;
    public BankAccount() { }
    public void setOwner(String owner){
      this.owner = owner;
    }
    public String getOwner(){ return owner; }
}
public class Test{
    public static void main(String args[]){
      BankAccount acc1 = new BankAccount();
      BankAccount acc2 = new BankAccount();
      acc1.setOwner("Hoa");
      acc2.setOwner("Hong");
      System.out.println(acc1.getOwner() + " " +
                         acc2.getOwner());
}
```

Nội dung

1. Phương thức khởi tạo
2. Các loại phương thức khởi tạo
3. Khai báo và khởi tạo đối tượng
4. Sử dụng đối tượng
5. **Quản lý bộ nhớ và so sánh đối tượng**
6. Hủy bỏ đối tượng
7. Ví dụ và bài tập

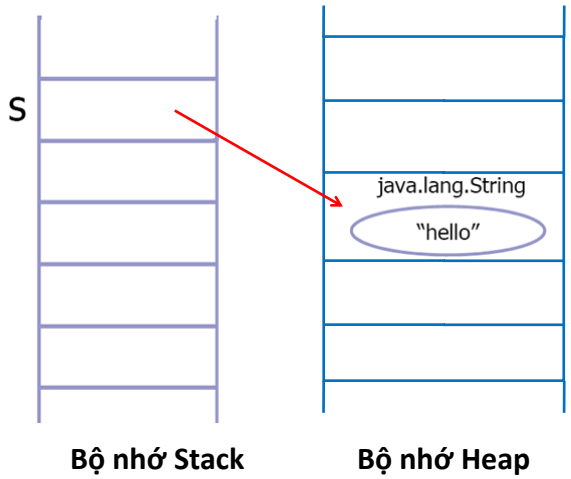
5. Quản lý bộ nhớ và so sánh đối tượng

- Java không sử dụng con trỏ nên các địa chỉ bộ nhớ không thể bị ghi đè lên một cách ngẫu nhiên hoặc cố ý.
- Các vấn đề định vị và tái định vị bộ nhớ, quản lý bộ nhớ do JVM kiểm soát, hoàn toàn trong suốt với lập trình viên.
- Lập trình viên không cần quan tâm đến việc ghi dấu các phần bộ nhớ đã cấp phát trong heap để giải phóng sau này.

Bộ nhớ Heap

```
String s = new String("hello");
```

- s là biến tham chiếu, lưu trên Stack
- Giá trị của s là địa chỉ của vùng nhớ Heap lưu trữ đối tượng s tham chiếu tới
- Bộ nhớ Heap sử dụng để ghi thông tin được tạo bởi toán tử **new**.

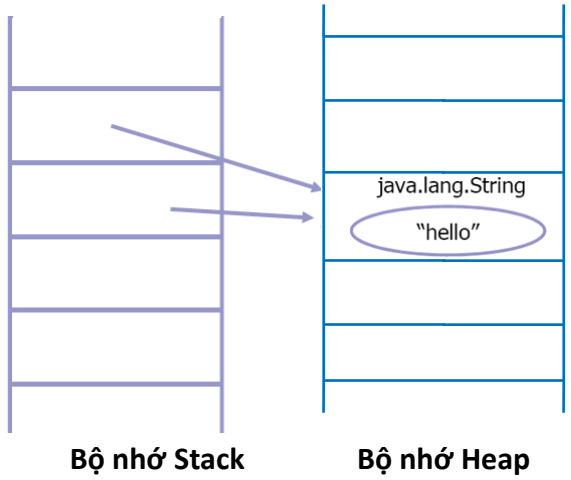


33

Bộ nhớ Heap

```
String s = new String("hello");
String t = s;
```

- s và t cùng tham chiếu tới một đối tượng (cùng trỏ tới một vùng nhớ trên heap)

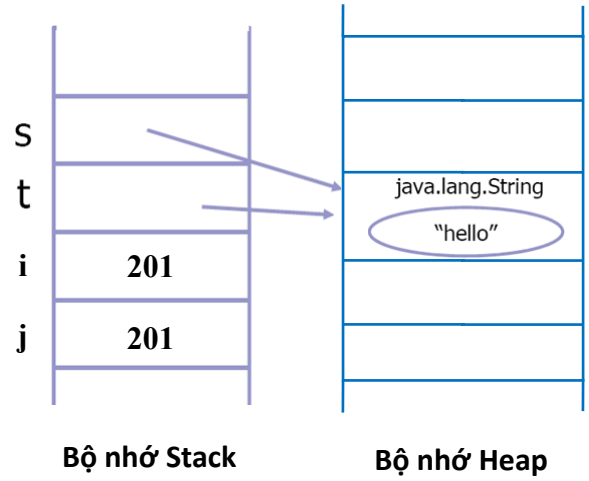


34

Bộ nhớ Stack

```
String s = new String("hello");
String t = s;
int i = 201;
int j = i;
```

- Giá trị cục bộ trong bộ nhớ Stack được sử dụng như con trỏ tham chiếu tới Heap
- Giá trị của dữ liệu nguyên thủy được ghi trực tiếp trong Stack



So sánh đối tượng

- Đối với các kiểu dữ liệu nguyên thủy, toán tử == kiểm tra xem chúng có giá trị bằng nhau hay không
- Ví dụ:

```
int a = 1;
int b = 1;
if (a==b)... // true
```

So sánh đối tượng (2)

- Đối với các đối tượng, toán tử == kiểm tra xem hai đối tượng có đồng nhất hay không, (có cùng tham chiếu đến một đối tượng hay không)
- Ví dụ:

a và b tham chiếu tới 2 đối tượng khác nhau

```
Employee a = new Employee(1);
Employee b = new Employee(1);
if (a==b)... // false
```

a và b cùng tham chiếu tới 1 đối tượng

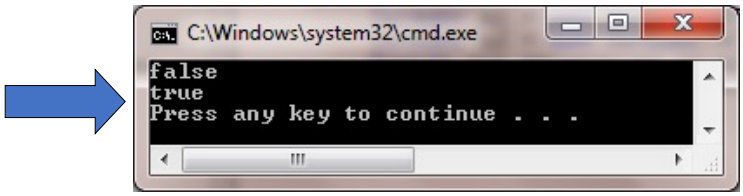
```
Employee a = new Employee(1);
Employee b = a;
if (a==b)... // true
```


So sánh đối tượng (3)

- Phương thức equals
  - Đối với kiểu dữ liệu nguyên thủy: Không tồn tại.
  - Đối với các đối tượng: Bất kỳ đối tượng nào cũng có phương thức này, dùng để so sánh giá trị của đối tượng
  - Phương thức equals kế thừa từ lớp Object (chi tiết xem bài Kết tập và kế thừa)
  - Cài đặt mặc định của phương thức equals là như toán tử ==. Cần cài đặt lại để so sánh 2 đối tượng dựa trên từng thuộc tính

Ví dụ == và equals – Lớp Integer

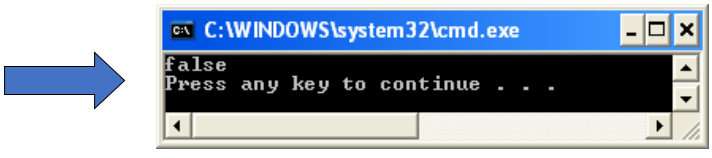
```
public class Equivalence {
        public static void main(String[] args) {
                Integer n1 = new Integer(47);
                Integer n2 = new Integer(47);
                System.out.println(n1 == n2);
                System.out.println(n1.equals(n2));
        }
}
```



Lớp Integer (lớp cung cấp trong Java SDK) đã cài đặt lại phương thức equals của lớp Object, nên n1.equals(n2) trả về true

Ví dụ sử dụng equals với lớp tự viết

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



Lớp Value (LTV tự viết) chưa cài đặt lại phương thức equals của lớp Object, nên v1.equals(v2) trả về false, giống như toán tử ==

Nội dung

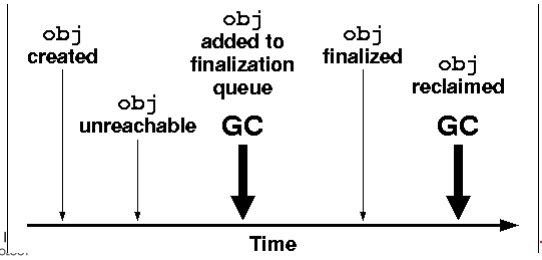
1. Phương thức khởi tạo
2. Các loại phương thức khởi tạo
3. Khai báo và khởi tạo đối tượng
4. Sử dụng đối tượng
5. Quản lý bộ nhớ và so sánh đối tượng
6. **Hủy bỏ đối tượng**
7. Ví dụ và bài tập

6. Hủy bỏ đối tượng

- Trong C#, C++:
  - Sử dụng phương thức hủy (destructor)
  - Phương thức hủy là phương thức tự động được gọi trước khi đối tượng được hủy
  - Phương thức hủy thường dùng để dọn dẹp bộ nhớ, thu hồi tài nguyên (VD đối tượng khi hoạt động cần truy cập tới file/CSDL, cấp phát bộ nhớ động)
- Trong Java:
  - Không có khái niệm phương thức hủy
  - Sử dụng phương thức finalize()

Phương thức void finalize()

- Lớp nào cũng có phương thức finalize() – được thực thi ngay lập tức khi quá trình thu gom xảy ra
- Thường chỉ sử dụng cho các trường hợp đặc biệt để "tự dọn dẹp" các tài nguyên sử dụng khi đối tượng được gc giải phóng
  - Ví dụ cần đóng các socket, file,... nên được xử lý trong luồng chính trước khi các đối tượng bị ngắt bỏ tham chiếu.
- Có thể coi là phương thức hủy (destructor) của lớp mặc dù Java không có khái niệm này.



43

Bộ thu gom rác (Garbage Collector)

- Một tiến trình chạy ngầm gọi đến bộ "thu gom rác" để phục hồi lại phần bộ nhớ mà các đối tượng không tham chiếu đến (tái định vị)
- Các đối tượng không có tham chiếu đến được gán null.
- Bộ thu gom rác định kỳ quét qua danh sách các đối tượng của JVM và phục hồi các tài nguyên của các đối tượng không có tham chiếu.

44

Bộ thu gom rác (2)

- JVM quyết định khi nào thực hiện thu gom rác:
  - Thông thường sẽ thực thi khi thiếu bộ nhớ
  - Tại thời điểm không dự đoán trước
- Không thể ngăn quá trình thực hiện của bộ thu gom rác nhưng có thể yêu cầu thực hiện sớm hơn:

```
System.gc(); hoặc Runtime.gc();
```


Nội dung

1. Phương thức khởi tạo
2. Các loại phương thức khởi tạo
3. Khai báo và khởi tạo đối tượng
4. Sử dụng đối tượng
5. Quản lý bộ nhớ và so sánh đối tượng
6. Hủy bỏ đối tượng
7. **Ví dụ và bài tập**

51