TRƯỜNG ĐHBK HÀ NỘI
VIỆN CNTT & TT

-----oOo-----

BÀI THỰC HÀNH

MÔN

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Mã HP: IT3600

Họ tên sinh viên : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Mã sinh viên : _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Nhóm : _

HÀ NỘI - 2017

MỤC LỤC

CÁC QUI ĐỊNH

Trong quá trình thực hành môn Lập trình hướng đối tượng, sinh viên phải chú ý đọc kỹ, nhớ và tuân theo các điều sau:

1. Giới thiệu quyển Bài thực hành Lập trình Hướng đối tượng

Đây là quyển Bài thực hành cho môn **Lập trình hướng đối tượng** cho các lớp CN-CNTT.

Mọi sinh viên học môn Lập trình Hướng đối tượng **đều phải có quyển Bài thực hành Lập trình Hướng đối tượng này và phải mang theo khi đi thực hành** (mỗi giảng đường sẽ được phát một quyển để photo, mỗi sinh viên một bản), phải đọc kỹ các phần hướng dẫn và làm đầy đủ bài tập theo yêu cầu.

- Các bài ***Ví dụ tham khảo*** là các ví dụ làm mẫu cho sinh viên, cần đọc và tìm hiểu kỹ trước khi làm các bài tập khác. Các bài này không cần làm vào vở hay thực hành trên máy.
- Các bài tập ***Làm vào vở ở nhà*** là các bài tập bắt buộc phải làm vào **Vở Bài tập Lập trình Hướng đối tượng riêng** ở nhà trước buổi thực hành và sẽ chạy thử trên máy tính trong buổi thực hành.
- Các bài tập ***Làm trong giờ thực hành*** là bắt buộc phải làm và chạy thử trên máy tính ngay trong giờ thực hành.
- Các bài tập ***Nâng cao*** là không bắt buộc, sinh viên có thể làm thêm sau khi đã hoàn thành các bài tập bắt buộc.

Các bài tập trong quyển này chỉ là một phần trong các bài tập sinh viên cần làm. Do vậy, ngoài các bài tập trong quyển này, sinh viên vẫn phải làm thêm các bài tập khác theo hướng dẫn của giáo viên dạy lý thuyết môn này.

Để hoàn thiện hơn nữa tài liệu này, trong quá trình làm thực hành, nếu thấy tài liệu này có gì sai sót hoặc có thể cải tiến, sinh viên có thể góp ý trực tiếp cho các thầy cô hướng dẫn hay về thầy Trịnh Thành Trung – BM CNPM - Viện CNTT-TT, Đại học Bách Khoa Hà Nội. Email: trungtt@soict.hust.edu.vn

2. Địa điểm thực hành

Phòng thí nghiệm: B1-701 (Bộ môn Công nghệ phần mềm)

3. Lịch thực hành

- Có tất cả **5** bài thực hành, thực hiện trong **3** buổi, mỗi buổi **3** bài **240** phút.
- Các nhóm thí nghiệm đã đăng ký từ đầu kỳ trên sis. Sinh viên phải biết mình thuộc nhóm số mấy và theo dõi lịch thực hành của nhóm đó. Sinh viên phải đi thực hành đúng thứ và kíp được phân cho nhóm của mình.
- Lịch thực hành cơ bản không thay đổi. Tuy nhiên thực tế có thể có các thay đổi nhỏ do các nguyên nhân bất khả kháng như: mất điện, ngày lễ, ngày thi học kỳ v.v. . . Khi đó sẽ có lịch thay đổi cho phù hợp và sẽ thông báo kịp thời. Các bài thực hành bị hoãn trong tuần nào sẽ cố gắng bù ngay trong tuần đó hoặc tuần tiếp theo.

4. Trình tự thực hiện 1 bài thực hành

1) Đầu kíp thực hành, cán bộ hướng dẫn bàn giao tình trạng máy trong phòng với đại diện sinh viên. Sinh viên có thể sử dụng máy tính cá nhân nếu không muốn sử dụng máy trong phòng thí nghiệm

2) Cán bộ hướng dẫn điểm danh và kiểm tra thẻ sinh viên theo danh sách của phòng Đào tạo. Cán bộ hướng dẫn kiểm tra vở bài tập nếu hôm đó có bài tập làm vào vở.

3) Trước khi sinh viên bật máy cán bộ nhắc nhở sinh viên toàn phòng máy đọc phần hướng dẫn ở đầu mỗi bài thực hành của mỗi bài để biết trọng tâm của bài thực hành và các chú ý khi làm bài thực hành.

4) Sinh viên chỉ thực hành theo nội dung trong quyển **Bài thực hành** và theo hướng dẫn của cán bộ. Nếu thực hành xong các bài trong quyển **Bài thực hành** có thể làm thêm các bài tập trong quyển lý thuyết hay thầy dạy lý thuyết cho.

5) Cán bộ hướng dẫn theo dõi, giúp đỡ, nhắc nhở chung cho sinh viên toàn phòng máy. Chú ý nhắc chung các lỗi hay gặp cho cả phòng và nhắc sinh viên phải xem kỹ các lời hướng dẫn, các ví dụ.

6) Trước khi hết giờ thực hành cán bộ hướng dẫn nhắc nhở sinh viên tắt máy, xếp ghế gọn gàng, bàn giao máy và trật tự ra về ngay.

7) Báo cáo kết quả thực hành phải được nộp trên hệ thống hỗ trợ học trực tuyến theo đúng thời hạn quy định của mỗi buổi thực hành.

5. Nội quy thực hành

- Khi đến thực hành sinh viên phải mang theo:
  - ***Thẻ Sinh Viên*** hoặc ***CMT***
  - Quyển ***Bài tập thực hành Lập trình Hướng đối tượng*** này
  - Vở ***Bài tập thực hành Lập trình Hướng đối tượng*** đủ làm đầy đủ các bài tập theo yêu cầu. Vở ***Bài tập thực hành Lập trình Hướng đối tượng*** phải được dành riêng một quyển, không chung với các môn khác hay chung với vở lý thuyết. Vở phải ghi rõ họ tên sinh viên và nhóm thực hành. Nếu vở không nghiêm túc, không đầy đủ bài tập sẽ bị trừ điểm thực hành.
- Đi thực hành đúng giờ và giữ trật tự, kỷ luật theo qui chế học tập chung của trường ĐHBK Hà Nội và nội quy của Phòng máy trong khi thực hành. ***Ai vi phạm sẽ bị trừ điểm thực hành và xử lý theo qui chế học tập của trường ĐHBK.***
- Đầu mỗi buổi thực hành, trước khi thao tác trên máy, phải ***đọc kỹ phần hướng dẫn*** của mỗi bài thực hành và ***nghe giáo viên hướng dẫn*** các điểm cần chú ý khi thực hành.
- Sinh viên chỉ thực hành theo nội dung trong quyển **Bài thực hành** và theo hướng dẫn của cán bộ. Nếu thực hành xong các bài trong quyển **Bài thực hành** có thể làm thêm các bài tập trong quyển lý thuyết hay thầy dạy lý thuyết cho.
- Sinh viên phải ***xem kỹ các ví dụ*** để tham khảo và lập trình theo.

6. Cách đánh giá tính điểm thực hành môn Lập trình Hướng đối tượng

- Điểm thực hành cho theo thang điểm **10**, mỗi bài thực hành tương ứng với **2** điểm thực hành.

- Mỗi bài thực hành sinh viên sẽ được tối đa **2** điểm thực hành nếu có mặt và về đúng giờ, nộp bài tập đúng hạn và làm bài tập trong phòng máy đầy đủ, không vi phạm nội quy thực hành. Nếu buổi thực hành nào sinh viên vi phạm một trong các điều trên thì buổi đó sẽ bị trừ điểm.
- Điểm thực hành sẽ được tích hợp cùng Điểm Kiểm Tra Giữa Kỳ để tính Điểm Quá Trình của môn học Lập trình Hướng đối tượng.
- Các vi phạm nghiêm trọng trong một buổi thực hành như mất trật tự, chơi trò chơi, nghịch làm hư hỏng thiết bị, máy móc sẽ bị trừ điểm thực hành và xử lý theo qui chế học tập của trường ĐHBK.
- ***Các qui định cụ thể về cho điểm, trừ điểm thực hành như sau:***
  - Vắng **1** buổi thực hành trừ **2** điểm thực hành. Vắng từ **3** buổi trở lên bị **0** điểm thực hành.
  - ***Đi muộn*** sau **15 phút** trừ **1** điểm, đi muộn sau **30** phút trừ **2** điểm.
  - Trong 1 buổi thực hành không làm bài tập ra vở trừ **2** điểm, làm thiếu bài tập hay làm ra giấy trừ **1** điểm.
  - Trong 1 buổi thực hành không mang quyển **Bài thực hành Lập trình Hướng đối tượng** này trừ **1** điểm.
  - Trong mỗi buổi thực hành không đọc các qui định, hướng dẫn, không lắng nghe giáo viên hướng dẫn, không xem các ví dụ bị trừ từ **1** đến **2** điểm.
  - Trong 1 buổi thực hành sinh viên vi phạm kỷ luật thực hành như mất trật tự, làm việc riêng, thực hành không đúng nội dung, bài tập làm trên máy sai, thiếu v.v … trừ từ **1** đến **2** điểm. Các trường hợp vi phạm kỷ luật nặng sẽ bị **0** điểm thực hành cho cả đợt thực hành và bị kỷ luật theo qui định của trường.
  - ***Có thể có kiểm tra đột xuất việc sinh viên làm bài tập trong một số buổi thực hành***. Giáo viên hướng dẫn có thể kiểm tra sinh viên làm bài tập trên máy. Nếu không làm nghiêm túc, làm thiếu bài tập hay làm sai nhiều có thể bị trừ từ **1** đến **2** điểm thực hành. Việc kiểm tra có thể được thực hiện trong ***một buổi thực hành bất kỳ, không báo trước***. Việc chọn ***lớp nào, buổi nào*** để kiểm tra là hoàn toàn ngẫu nhiên, không báo trước. Do vậy có thể có những lớp kiểm tra 2 lần hoặc có những lớp không kiểm tra.

Bài thực hành số 1

THIẾT LẬP MÔI TRƯỜNG LÀM VIỆC

A. HƯỚNG DẪN

1. Nghe giáo viên phổ biến nội quy và hướng dẫn cách thực hành.
2. Đọc kỹ **Lời nói đầu** của quyển bài tập này trước khi thực hành.
3. Thực hiện cài đặt các công cụ lập trình (nếu cần)
4. Làm các bài tập trong tài liệu thực hành
5. Nếu làm xong các bài tập và còn thời gian sinh viên có thể làm sang Bài thực hành số 2
6. Hết giờ thực hành sinh viên tắt máy, để gọn ghế, ra về trật tự.

B. VÍ DỤ VÀ BÀI TẬP

Bài 1.1: Cài đặt JDK

Mục đích: *Cài đặt bộ công cụ lập trình Java*

Yêu cầu: *Làm trên máy trong buổi thực hành*

Đối với hệ điều hành Windows:

- Download tại đây http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Sau đó làm theo hướng dẫn

Đối với hệ điều hành Mac, Linux

- Tìm và cài đặt gói **default-jdk** hoặc **openjdk-7-jdk**

Bài 1.2: Viết ứng dụng Java đơn giản

Mục đích: *Bảo đảm việc máy tính đã đủ điều kiện lập trình Java*

Yêu cầu: *Làm trên máy trong buổi thực hành*

Sử dụng Notepad hoặc Notepad++ (https://notepad-plus-plus.org/) viết mã nguồn như sau

```
public class Hello {

   public static void main (String[] args) {
      System.out.println ("Hello World!");
   }
}
```

Lưu lại vào file ***Hello.java***.

Sử dụng Command Prompt (hoặc Terminal) biên dịch file ở trên dùng javac

```
javac Hello.java
```

File Hello.class sẽ được tạo ra. Để chạy, ta dùng lệnh

```
java Hello
```

Bài 1.3: Cài đặt các môi trường phát triển tích hợp (IDE)

Mục đích: *Cài đặt các môi trường phát triển tích hợp, giúp việc lập trình trở nên thuận tiện hơn*

Yêu cầu: *Làm trên máy trong buổi thực hành*

Sinh viên cài đặt một trong các IDE sau:

- NetBeans: Công cụ lập trình Java, PHP, C++ (https://netbeans.org/)
- Eclipse: Công cụ lập trình Java, PHP, C++ (https://www.eclipse.org/)
- IntelliJ: Công cụ lập trình Java (https://www.jetbrains.com/idea/)
    - Vào link https://www.jetbrains.com/student/ để đăng ký (Sử dụng email của trường)

Đối với hệ điều hành Windows, sinh viên có thể vào trang web download về và chạy installer.

Đối với hệ điều hành Mac và Linux, sinh viên có thể tìm tên các ứng dụng trong App Store tương ứng để cài đặt. Ngoài ra sinh viên cũng có thể vào trực tiếp trang web của các IDE và làm theo hướng dẫn.

Bài 1.4: Viết các chương trình Java cơ bản

Mục đích: *Làm quen với ngôn ngữ lập trình Java*

Yêu cầu: *Làm trên máy trong buổi thực hành*

1. Thực hành viết, dịch và chạy ví dụ sau:

```
// Vi du 3: HelloNameDialog.java
import javax.swing.JOptionPane;
public class HelloNameDialog{
 public static void main(String[] args){
   String result;
   result = JOptionPane.showInputDialog("Hay nhap ten ban:");
   JOptionPane.showMessageDialog(null, "Xin chao "+ result + "!");
   System.exit(0);
 }
}
```

2. Viết chương trình in ra màn hình tam giác có chiều cao là 5 * như sau:

```
    *
   ***
  *****
 *******
*********
```

3. Thực hành viết, dịch và chạy ví dụ sau.

```
// Vi du 5: HienThiHaiSo.java
import javax.swing.JOptionPane;
public class HienThiHaiSo{
public static void main(String[] args){
   String strSo1, strSo2;
   String strHienThi = "Ban vua nhap ";

   strSo1 = JOptionPane.showInputDialog(null, "Hay nhap so thu 1: ",
"Nhap so thu nhat", JOptionPane.INFORMATION_MESSAGE);
   strHienThi += strSo1 + " va ";
   strSo2 = JOptionPane.showInputDialog(null, "Hay nhap so thu 2: ",
"Nhap so thu hai", JOptionPane.INFORMATION_MESSAGE);
   strHienThi += strSo2;

   JOptionPane.showMessageDialog(null,strHienThi, "Hien thi hai so",
                        JOptionPane.INFORMATION_MESSAGE);
   System.exit(0);
 }
}
```

4. Sửa ví dụ 4 viết chương trình tính và hiển thị tổng 2 số vừa nhập từ bàn phím.
Gợi ý: Khai báo thêm hai biến nguyên và thực hiện chuyển kiểu dữ liệu từ xâu ký tự sang số nguyên, sử dụng hàm Integer.parseInt(strSo1),…

5. Thực hành viết, dịch và chạy ví dụ sau:

```
import javax.swing.JOptionPane;
public class LuaChon{
 public static void main(String[] args){
      int iLuaChon;
      String strLuaChon;

      iLuaChon = JOptionPane.showConfirmDialog(null,
"Ban co muon chuyen sang ve hang nhat khong?");
```

```
        if (iLuaChon == JOptionPane.YES_OPTION)
              strLuaChon = "co";
        else
              strLuaChon = "khong";

        JOptionPane.showMessageDialog(null,"Ban da chon " + strLuaChon);
        System.exit(0);
    }
}
```

Sửa lệnh showConfirmDialog trong chương trình trên thành đoạn mã dưới đây. Quan sát kết quả và đưa ra nhận xét với các tham số và chồng phương thức của showConfirmDialog()

```
iLuaChon = JOptionPane.showConfirmDialog(null, "Co loi xay ra. Co muon
tiep tuc?", "Loi", JOptionPane.YES_NO_OPTION,
JOptionPane.ERROR_MESSAGE);
```

Bài 1.5: Viết các chương trình Java cơ bản

Mục đích: *Làm quen với ngôn ngữ lập trình Java*

Yêu cầu: *Làm vào máy tính ở nhà và nộp bài tập lên hệ thống. Chạy thử trên máy tính trong buổi thực hành sau*

1. Viết chương trình nhập các số thực rồi tính tổng các số thực đó. Sau mỗi lần nhập một số thực, đều quay lại hỏi người dùng có nhập tiếp hay không, nếu có thì yêu cầu nhập tiếp. Nếu không thì dừng và đưa ra kết quả.

Gợi ý: Dùng do…while() và hàm Double.parseDouble(strDouble)

2. Viết chương trình nhập tháng và năm từ bàn phím. Sau đó đưa ra số ngày tương ứng của tháng thuộc năm đã nhập. Yêu cầu có kiểm tra điều kiện: tháng là số nguyên từ 1 đến 12. Năm là 1 số nguyên > 0. Nếu người dùng nhập sai thì yêu cầu nhập lại.

Bài thực hành số 2

XÂY DỰNG LỚP VÀ SỬ DỤNG ĐỐI TƯỢNG

A. HƯỚNG DẪN

1. Bài tập trong bài thực hành này ứng với phần lý thuyết Bài 3: Xây dựng lớp và Bài 4: Các kỹ thuật xây dựng lớp
2. Trọng tâm phần này là nắm được cách thiết kế các lớp. Xây dựng các lớp và sử dụng đối tượng ở trong Java
3. Nếu làm xong các bài tập và còn thời gian sinh viên có thể làm các bài tập về nhà
4. Hết giờ thực hành sinh viên tắt máy, để gọn ghế, ra về trật tự.

B. LÝ THUYẾT

Một lớp là một thiết kế (blueprint) hay mẫu (prototype) cho các đối tượng cùng kiểu

- Ví dụ: lớp XeDap là một thiết kế chung cho nhiều đối tượng xe đạp được tạo ra

Lớp định nghĩa các thuộc tính và các phương thức chung cho tất cả các đối tượng của cùng một loại nào đó

Một đối tượng là một thể hiện cụ thể của một lớp.

- Ví dụ: mỗi đối tượng xe đạp là một thể hiện của lớp XeDap

Mỗi thể hiện có thể có những thuộc tính thể hiện khác nhau

- Ví dụ: một xe đạp có thể đang ở bánh răng thứ 5 trong khi một xe khác có thể là đang ở bánh răng thứ 3.

Tính đóng gói

Ẩn đi các chi tiết thực hiện bên trong, cung cấp cho các lớp bên ngoài một giao diện

- Các thành phần thuộc tính là private
- Xây dựng các phương thức getter và setter để truy cập vào dữ liệu

Nạp chồng

Xây dựng các phương thức trong cùng một lớp cùng tên nhưng khác danh sách tham số

- Mục đích: Mô tả bản chất công việc giống nhau

C. VÍ DỤ VÀ BÀI TẬP

Bài 2.1: Xây dựng lớp đại diện cho tài khoản ngân hàng

Mục đích: *Hiểu các khái niệm về lớp và đối tượng. Xây dựng thuộc tính cho lớp*

Yêu cầu: *Làm trên máy trong buổi thực hành*

Hướng dẫn: Tạo một file Java mới đặt tên là **TaiKhoan.java** với nội dung

```
class Account {
	// Member variable
	String name; // Account name
	int balance; // Balance

	// Value setting method
	void setData(String pName, int pBalance) {
		name = pName;
		balance = pBalance;
	}
	// display() method without argument
	void display() {
		System.out.println("There is no argument. ");
		System.out.print("Account name :" + name);
		System.out.println("\tBalance :" + balance + "VND");
	}
}
```

Trong lớp chứa hàm main(), tạo và sử dụng các đối tượng tài khoản ngân hàng tương ứng

```
class AccountExample {
	public static void main(String args[]) {
		// Creation of account object
		Account obj = new Account();
		// Sending message (Method call)
		obj.setData("Minh Nam", 100000);
		obj.display();
	}
}
```

Bài 2.2: Viết các phương thức rút tiền và gửi tiền cho tài khoản ngân hàng

Mục đích: *Nắm được cách xây dựng phương thức cho lớp*

Yêu cầu: *Làm trên máy trong buổi thực hành*

Hướng dẫn: Trong lớp TaiKhoan, bổ sung phương thức gửi tiền

```
void deposit (long money){
	balance +=money;
}
```

Viết tương tự cho phương thức rút tiền. Lưu ý phải kiểm tra số dư còn đủ để rút tiền không.

Trong phương thức main, để gọi đến phương thức rút tiền của tài khoản obj đã tạo, ta dùng

```
// Deposit processing
obj.deposit(20000);
```

Bài 2.3: Xây dựng phương thức khởi tạo mặc định và phương thức khởi tạo có 2 tham số (tên, số dư tài khoản) cho tài khoản ngân hàng

Mục đích: *Nắm được cách xây dựng phương thức khởi tạo cho lớp*

Yêu cầu: *Làm trên máy trong buổi thực hành*

Hướng dẫn

```
class Account {
      ...
      // Constructor without argument
      Account() {
            name = "Unsigned";
            balance = 0;
      }

      // Constructor with 2 arguments
      Account(String pName, int pBalance) {
            name = pName;
            balance = pBalance;
      }
      ...
}
```

Bài 2.4: Ẩn các thành phần dữ liệu, sử dụng các phương thức getter và setter để gán và lấy dữ liệu cho tài khoản ngân hàng

Mục đích: *Nắm được cách sử dụng tính đóng gói (encapsulation)*

Yêu cầu: *Làm trên máy trong buổi thực hành*

Hướng dẫn

```
class Account {
      // Member variable
      private String name;
      private int balance; // Balance (private Specification)

      pubic Account() {
            name = "Unsigned";
            balance = 0;
      }

      // Constructor with 2 arguments
      public Account(String pName, int pBalance) {
```

```
            name = pName;
            balance = pBalance;
        }

        // Set Value Method
        public void setData(String pName, int pBalance) {
            name = pName;
            balance = pBalance;
        }
        // Refer value (account name)  Method
        public String getName() {
            return name;
        }
        // Refer value (balance) Method
        public int getBalance() {
            return balance;
        }
        // Display value Method
        public void display() {
            System.out.print("Account name :" + name);
            System.out.println("\tBalance :" + balance + "USD");
        }
    }
```

Bài 2.5: Viết phương thức `display(String currency)` nạp chồng phương thức `display()`, thực hiện công việc: Nếu currency nhập vào là "VND" thì sẽ in ra balance được tính bằng đồng Việt Nam, nếu currency nhập vào là "GBP" thì sẽ in ra balance được tính bằng bảng Anh... (tỉ giá quy đổi tính theo ngày làm bài thực hành)

Mục đích: *Nắm được cách sử dụng kỹ thuật nạp chồng (overloading)*

Yêu cầu: *Làm trên máy trong buổi thực hành*

Hướng dẫn: Viết các phương thức cùng tên và khác tham số, nội dung của phương thức có thể viết khác nhau tùy vào loại tham số đưa vào phương thức.

Bài 2.6: Xây dựng lớp phương tiện đi lại (Vehicle) với các thuộc tính: *X, Y, angle, velocity, turnSpeed*, các phương thức:

- *ahead*(): tiến về phía trước với vận tốc velocity
- *turnLeft*(): quay sang trái với tốc độ turnSpeed
- *turnRight*(): ngược lại turnLeft
- *display*(): hiện thị tọa độ và góc hiện tại của xe

Mục đích: *Ứng dụng các kỹ thuật xây dựng lớp đã học*

Yêu cầu: *Làm vào máy tính ở nhà và nộp bài tập lên hệ thống. Chạy thử trên máy tính trong buổi thực hành sau*

Hướng dẫn: Các phương thức *ahead(), turnLeft(), turnRight()* mỗi lần được gọi sẽ thay đổi các thuộc tính vị trí và góc của xe theo tốc độ di chuyển *(velocity)* và tốc độ quay *(turnSpeed)* tương ứng. Ví dụ muốn cho đối tượng xe **car01** đi về phía trước 50 bước, ta có thể gọi đến

```
for(int i = 0; i < 50; i++)
    car01.ahead();
```

Bài 2.7: Bổ sung các phương thức sau cho lớp Vehicle:

- *ahead*(double distance): tiến từ từ về phía trước đến vận tốc velocity cho đến khi đi đủ distance (Gợi ý: Cần khai báo thêm thuộc tính gia tốc acceleration)
- *turnLeft*(double degree): quay sang trái với tốc độ turnSpeed một góc degree
- *turnRight*(double degree): ngược lại turnLeft

Mục đích: *Ứng dụng các kỹ thuật xây dựng lớp đã học*
Yêu cầu: *Nâng cao. Làm vào máy tính ở nhà.*

Bài thực hành số 3

KẾT TẬP VÀ KẾ THỪA

A. HƯỚNG DẪN

1. Bài tập trong bài thực hành này ứng với phần lý thuyết Bài 4: Các kỹ thuật xây dựng lớp và Bài 5: Kế thừa
2. Trọng tâm phần này là nắm các kỹ thuật Kết tập và Kế thừa khi xây dựng lớp
3. Nếu làm xong các bài tập và còn thời gian sinh viên có thể làm các bài tập về nhà
4. Hết giờ thực hành sinh viên tắt máy, để gọn ghế, ra về trật tự.

B. LÝ THUYẾT

Kết tập

Là mối quan hệ "chứa" hay "là một phần" (ví dụ: Tứ giác chứa 4 điểm, điểm là một phần của tứ giác)

- Tái sử dụng mã nguồn thông qua đối tượng

Kế thừa

Phát triển lớp cũ bằng cách xây dựng lớp mới, kế thừa các phương thức và thuộc tính của lớp cũ

- Cú pháp: Lớp Mới extends Lớp Cũ
- Mối quan hệ: "là" (is-a) (ví dụ: Hình vuông là một tứ giác, Sinh viên là một người)

C. VÍ DỤ VÀ BÀI TẬP

Bài 3.1: Xây dựng lớp Ngân Hàng (Bank) chứa các tài khoản (Account).

- **Xây dựng phương thức thêm và xóa tài khoản ngân hàng.**
- **Viết phương thức in thông tin tất cả các tài khoản ngân hàng. Phương thức này sẽ in ra tên chủ tài khoản và số dư của tài khoản.**
- **Viết phương thức tính tổng tiền của tất cả các tài khoản ngân hàng.**

Mục đích: *Nắm được khái niệm về kỹ thuật kết tập và ứng dụng trong chương trình*
Yêu cầu: *Làm trên máy trong buổi thực hành*

Hướng dẫn: Tạo một file Java mới đặt tên là **Bank.java**, tạo một lớp **Bank**, trong đó khai báo một mảng (hoặc danh sách) các đối tượng **Account**
Các phương thức in thông tin và tính tổng tiền gọi đến các phương thức in thông tin và lấy số dư đã viết trong lớp **Account**

Bài 3.2: Xây dựng lớp Tài Khoản Tiết Kiệm (SavingAccount) kế thừa lớp Tài Khoản (Account), biết

- **Lớp Tài Khoản Tiết Kiệm có thêm thuộc tính lãi suất và phương thức tính lãi.**
- **Phương thức tính lãi sẽ được gọi định kỳ mỗi năm một lần. Mỗi lần được gọi, số dư của tài khoản sẽ được cộng thêm một khoản tiền theo lãi suất của tài khoản**

Mục đích: *Nắm được khái niệm về kỹ thuật kế thừa và ứng dụng trong chương trình*
Yêu cầu: *Làm trên máy trong buổi thực hành*

Hướng dẫn: Tạo một file Java mới đặt tên là **SavingAccount.java**, tạo một lớp mới đặt tên là **SavingAccount** kế thừa từ lớp **Account**

```
class SavingAccount extends Account {
      Account[] accountList;
}
```

Các phương thức in thông tin và tính tổng tiền gọi đến các phương thức in thông tin và lấy số dư đã viết trong lớp **Account**

Bài 3.3:

A) Xây dựng các lớp con kế thừa từ lớp Vehicle (ví dụ Car, Bike, Truck, Motobike,...)

- **Mỗi lớp con này có một phương thức `move()`: thực hiện di chuyển theo một số cách khác nhau (ví dụ đi một đoạn ngắn rồi rẽ trái hoặc rẽ phải rồi đi một đoạn dài...)**

B) Xây dựng lớp Map kết tập nhiều đối tượng là các lớp con của lớp Vehicle ở trên.

- **Viết phương thức `update()` cho lớp Map, gọi đến phương thức `move()` của tất cả các đối tượng xe, sau đó gọi đến phương thức `display()` để hiển thị tọa độ hiện tại của các xe đó.**

Mục đích: *Ứng dụng các kỹ thuật xây dựng lớp đã học*
Yêu cầu: *Làm vào máy tính ở nhà và nộp bài tập lên hệ thống. Chạy thử trên máy tính trong buổi thực hành sau*

Bài thực hành số 4

GHI ĐÈ, LỚP TRỪU TƯỢNG & GIAO DIỆN

A. HƯỚNG DẪN

1. Bài tập trong bài thực hành này ứng với phần lý thuyết Bài 7: Ghi đè, lớp trừu tượng & giao diện
2. Trọng tâm phần này là nắm các kỹ thuật Ghi đè, lớp trừu tượng và giao diện trong kế thừa
3. Nếu làm xong các bài tập và còn thời gian sinh viên có thể làm sang Bài thực hành số 5
4. Hết giờ thực hành sinh viên tắt máy, để gọn ghế, ra về trật tự.

B. LÝ THUYẾT

Ghi đè

Phương thức ở lớp con trùng tên, trùng danh sách tham số với phương thức ở lớp cha.

- Trình biên dịch sẽ quyết định gọi phương thức nào tùy vào kiểu đối tượng được gọi

Lớp trừu tượng

Là một lớp không thể thể hiện hóa trực tiếp

- Cú pháp: sử dụng từ khóa **abstract**
- Một lớp trừu tượng phải có ít nhất một phương thức trừu tượng
- Các lớp con kế thừa phải có nhiệm vụ ghi đè phương thức trừu tượng này
- Nếu một lớp đã có một hoặc nhiều phương thức trừu tượng thì bắt buộc phải là lớp trừu tượng

Giao diện

Là một cấu trúc quy định những phương thức mà lớp thực thi nó phải thực hiện được.

- Cú pháp: sử dụng từ khóa **interface**
- Chỉ chứa các chữ ký phương thức mà không có phần thực thi.
- Để thực thi: sử dụng từ khóa **implements**
- Lớp thực thi giao diện phải cài đặt tất cả các phương thức khai báo trong giao diện, nếu không phải là lớp trừu tượng

C. VÍ DỤ VÀ BÀI TẬP

Bài 4.1: Lớp Tài khoản (Account) sửa thành lớp trừu tượng. Phương thức rút tiền và gửi tiền trở thành phương thức trừu tượng. Chỉ ra kết quả sau khi thực hiện các công việc sau

1. **Tạo một đối tượng Account và gọi đến các phương thức rút tiền và gửi tiền**
2. **Tạo một đối tượng SavingAccount và gọi đến các phương thức rút tiền và gửi tiền**

Mục đích: *Nắm được khái niệm về lớp trừu tượng trong Lập trình hướng đối tượng*
Yêu cầu: *Làm trên máy trong buổi thực hành*

Bài 4.2: Xây dựng các lớp sau kế thừa từ lớp Account:

1. **Xây dựng lớp Tài khoản vãng lai (CheckingAccount)** kế thừa lớp Tài khoản, có các phương thức rút tiền và gửi tiền. Phương thức rút tiền không cho phép rút quá số dư trong tài khoản. Tài khoản vãng lai không có lãi suất.
2. **Sửa lại lớp Tài khoản tiết kiệm (SavingAccount) ở bài trước.** Lớp này thực thi giao diện Tài khoản sinh lãi. Bổ sung thêm thuộc tính Ngày tạo tài khoản (dateCreated) cho lớp Tài khoản tiết kiệm. Sửa lại phương thức rút tiền và gửi tiền sao cho việc rút tiền và gửi tiền chỉ có thể thực hiện được vào đúng ngày tạo tài khoản hàng năm.
3. **Xây dựng lớp Tài khoản tín dụng (CreditAccount)** kế thừa lớp Tài khoản. Bổ sung thêm thuộc tính Hạn mức (limit), lãi suất gửi (debitInterest) và lãi suất ghi nợ (creditInterest) và Ngày tạo tài khoản (dateCreated). Phương thức rút tiền cho phép người dùng rút quá số dư trong tài khoản nhưng không vượt quá hạn mức. Phương thức tính lãi của tài khoản tín dụng phụ thuộc vào số dư trong tài khoản là lớn hơn hay nhỏ hơn 0. Nếu số dư lớn hơn 0 thì khi tính lãi, số dư sẽ được cộng thêm theo lãi suất gửi, còn nếu số dư nhỏ hơn 0 thì số dư sẽ bị tính thêm theo lãi suất ghi nợ.

Mục đích: *Nắm được khái niệm về lớp trừu tượng trong Lập trình hướng đối tượng*
Yêu cầu: *Làm trên máy trong buổi thực hành*

Bài 4.3: Tìm hiểu tài liệu về lập trình Robocode trong Phụ lục 1, áp dụng vào cài đặt trong NetBeans hoặc Eclipse. Xây dựng một xe tăng có khả năng di chuyển đơn giản.

Mục đích: *Tìm hiểu cách thức lập trình với các thư viện của Java.*
Yêu cầu: *Làm vào máy tính ở nhà và nộp bài tập lên hệ thống.*

Hướng dẫn:

Thêm thư viện Robocode vào NetBeans

Ta *chọn New Project* như bình thường, sau đó vào *Properties* rồi chọn *Add JAR*. Tiếp theo đó tìm đến thư mục **libs** của Robocode và chọn **robocode.jar**

Để kết hợp với Javadocs của Robocode, ấn vào *Edit* và chọn thư mục Javadoc của Robocode
Đối với Eclipse làm tương tự.

Viết mã nguồn cho chiếc xe tăng đầu tiên
Chọn *New Class* như bình thường. Lớp này kế thừa từ **robocode.Robot** hoặc có thể **`import robocode.*`** và kế thừa lớp **Robot**.
Sau đó ta có thể ghi đè (override) lại các phương thức của lớp **Robot**.
(xem thêm tài liệu ở Phụ lục 1)

Viết mã nguồn

```
public class MyFirstRobot extends Robot {
    public void run() {
        while (true) {
            ahead(100);
            turnGunRight(360);
            back(100);
            turnGunRight(360);
        }
    }
    public void onScannedRobot(ScannedRobotEvent e) {
        fire(1);
    }
}
```

Chạy thử robot trong Robocode
Chọn menu *Options* | *Preferences*, vào tab *Development* rồi add thư mục đến gói chứa các package của mình (thư mục **classes** trong NetBeans hoặc thư mục **bin** trong Eclipse).

Bài 4.4: Tìm hiểu về các phương thức quan trọng trong lớp robocode.Robot trong Phụ lục 1. Xây dựng một xe tăng có khả năng đuổi theo bắn một xe tăng khác
Mục đích: *Tìm hiểu cách thức lập trình với các thư viện của Java.*
Yêu cầu: *Làm vào máy tính ở nhà và nộp bài tập lên hệ thống.*

Hướng dẫn:
Dùng phương thức *getBearing()* và *ahead()*

```
public class MyFirstRobot extends Robot {
    public void run() {
       setAdjustRadarForGunTurn(true);
        while (true) {
            // Quay radar đến lúc phát hiện kẻ địch
            turnRadarRight(10000);
        }
```

```
    }

    public void onScannedRobot(ScannedRobotEvent e) {
            turnRight(e.getBearing()); // Quay về phía địch
            fire(3); // Bắn về phía kẻ địch
            ahead(e.getDistance()); // Tiến về phía địch
    }
}
```

Bài 4.5: Tìm hiểu về các phương thức quan trọng trong lớp robocode.Robot trong Phụ lục 1. Xây dựng một xe tăng có khả năng tuần tra theo một đường đi có sẵn, bắn khi gặp kẻ địch.

Mục đích: *Tìm hiểu cách thức lập trình với các thư viện của Java.*

Yêu cầu: *Nâng cao. Làm vào máy tính ở nhà.*

Bài thực hành số 5

ĐA HÌNH

A. HƯỚNG DẪN

1. Bài tập trong bài thực hành này ứng với phần lý thuyết Bài 8. Đa hình
2. Trọng tâm phần này là nắm các kỹ thuật Ghi đè, lớp trừu tượng và giao diện trong kế thừa
3. Nếu làm xong các bài tập và còn thời gian sinh viên có thể làm các bài tập về nhà
4. Hết giờ thực hành sinh viên tắt máy, để gọn ghế, ra về trật tự.

B. LÝ THUYẾT

Up-casting và down-casting

Chuyển đổi các kiểu dữ liệu đối tượng: nhìn nhận một đối tượng như kiểu của các lớp nằm trên cùng một cây phân cấp kế thừa

- Up-casting: đi lên trên cây phân cấp kế thừa, nhìn nhận một đối tượng của lớp cơ sở như là đối tượng của một lớp dẫn xuất. Java tự động chuyển đổi
- Down-casting: đi xuống dưới cây phân cấp kế thừa, nhìn nhận một đối tượng của lớp dẫn xuất như là đối tượng của một lớp cơ sở. Java không tự động chuyển đổi, phải ép kiểu.

Đa hình

Thực hiện cùng một công việc theo nhiều cách khác nhau

- Đa hình phương thức: Gọi phương thức theo các cách khác nhau tùy theo tham số
- Đa hình đối tượng: Gọi phương thức theo các cách khác nhau tùy theo đối tượng được gọi

C. VÍ DỤ VÀ BÀI TẬP

Bài 5.1: Bổ sung phương thức trừu tượng `inThongTin()` trong lớp Tài khoản ngân hàng (Account), các lớp con của lớp Tài khoản ngân hàng sẽ ghi đè phương thức này và thực hiện công việc in ra các thông tin liên quan đến tài khoản ngân hàng. Sau đó tạo ra một đối tượng của lớp Tài khoản tiết kiệm (hoặc Tài Khoản Tín Dụng hoặc Tài khoản vãng lai) , upcasting lên lớp Tài khoản và gọi đến phương thức `inThongTin()`. Chỉ ra phương thức `inThongTin()` của lớp nào được gọi đến.

Mục đích: *Nắm được cơ chế up-casting và down-casting.*

Yêu cầu: *Làm trên máy trong buổi thực hành*

Hướng dẫn: Tạo một đối tượng của lớp SavingAccount (hoặc CheckingAccount hay CreditAccount) và gán cho một đối tượng Account

```
Account acc = new SavingAccount(); hoặc
SavingAccount savingAcc = new SavingAccount();
Account acc = savingAcc;
```

Bài 5.2: Bổ sung phương thức `inThongTinTaiKhoan()` trong lớp Ngân hàng, thực hiện công việc in thông tin của tất cả các tài khoản trong ngân hàng.
Viết trong mã nguồn phương thức main tạo ra một đối tượng ngân hàng và các đối tượng của các lớp Tài khoản tiết kiệm, Tài khoản vãng lai, Tài khoản tín dụng; sau đó thêm các tài khoản ngân hàng đó vào ngân hàng. Thực hiện phương thức `inThongTinTaiKhoan()` của đối tượng ngân hàng và xem kết quả.

Mục đích: *Hiểu cách ứng dụng tính đa hình trong Lập trình hướng đối tượng.*
Yêu cầu: *Làm trên máy trong buổi thực hành*

Bài 5.3: Tìm hiểu về lớp AdvancedRobot trong Robocode (xem Phụ lục 1). Viết lại xe tăng ở bài trước sử dụng việc kế thừa từ AdvancedRobot
Mục đích: *Tìm hiểu cách thức lập trình với các thư viện của Java.*
Yêu cầu: *Làm vào máy tính ở nhà và nộp bài tập lên hệ thống.*

Hướng dẫn:
Khi chúng ta viết mã nguồn cho các lớp robot nâng cao, thay vì việc gọi trực tiếp đến fire, ahead, turnRight… ta phải gọi đến các phương thức set* tương ứng. Khi chúng ta muốn robot thực hiện tất cả các yêu cầu đó cùng lúc, ta bắt buộc phải gọi đến phương thức **execute()**

Ví dụ đối với xe tăng đã viết ở trong mục 4.4, ta có thể sửa lại như sau

```
public class NewBot extends AdvancedRobot {
    public void run() {
        setAdjustRadarForGunTurn(true);
        while (true) {
            setTurnRadarRight(10000);
            execute();
        }
    }
    public void onScannedRobot(ScannedRobotEvent e) {
        setTurnRight(e.getBearing());
        setFire(3);
        setAhead(e.getDistance());
    }
}
```

Bài 5.4: Tìm hiểu về lớp AdvancedRobot trong Robocode (xem Phụ lục 1). Viết mã nguồn cho xe tăng có khả năng tập trung bắn vào một mục tiêu.

Mục đích: *Tìm hiểu cách thức lập trình với các thư viện của Java.*

Yêu cầu: *Làm vào máy tính ở nhà và nộp bài tập lên hệ thống.*

Hướng dẫn:

Khi radar quét đến mục tiêu, ta sẽ muốn cố định hướng của radar theo đối phương. Để làm được việc này ta sẽ muốn gọi đến `setTurnRadarRight(e.getBearing());` Tuy nhiên `e.getBearing()` lại trả về góc của xe tăng với đối phương chứ không phải radar. Do đó, để cố định hướng của radar theo một mục tiêu ta phải xử lý được sự chênh lệch này, ta có thể viết

```
public void onScannedRobot(ScannedRobotEvent e) {
    // Lock on to our target (this time for sure)
    setTurnRadarRight(getHeading() - getRadarHeading() +
                      e.getBearing());
    //...
```

Ví dụ để viết một xe tăng tập trung bắn vào một mục tiêu

```
public class NewBot extends AdvancedRobot {
    public void run() {
        setAdjustRadarForGunTurn(true);
        setTurnRadarRight(1000); // Quét một lượt lần đầu
        execute();
        while (true) {
            // Dịch chuyển radar một chút để
            // gọi đến sự kiện onScannedRobot
            if (getRadarTurnRemaining() == 0) {
                setTurnRadarRight(1);
            }
            execute();
        }
    }
    public void onScannedRobot(ScannedRobotEvent e) {
        setTurnRight(e.getBearing()); // Quay về phía đối phương
        if (Math.abs(getTurnRemaining()) < 10) {
            // Tiến lại gần
            if (e.getDistance() > 200) {
                setAhead(e.getDistance() / 2);
            }
            // Nếu quá gần thì lùi lại
            if (e.getDistance() < 100) {
                setBack(e.getDistance() * 2);
            }
            setFire(3.0);
```

```
            }
            // Khóa radar
            setTurnRadarRight(getHeading() - getRadarHeading() +
                              e.getBearing());
        }
        public void onRobotDeath(RobotDeathEvent e) {
            // Sau khi đã tiêu diệt được đối thủ, tiếp tục quét robot
            setTurnRadarRight(1000);
        }
    }
```

Bài 5.5: Tìm hiểu về lớp AdvancedRobot trong Robocode (xem Phụ lục 1). Xây dựng một xe tăng có khả năng tìm và tiêu diệt mục tiêu cũng như tránh được sự tấn công của các xe tăng địch

Mục đích: *Tìm hiểu cách thức lập trình với các thư viện của Java.*

Yêu cầu: *Nâng cao. Làm vào máy tính ở nhà.*

TÀI LIỆU THAM KHẢO

1. **Giáo trình Lập trình Hướng đối tượng**. Viện CNTT-TT, ĐHBK HàNội. 2010.
2. **Giáo trình Lập trình Hướng đối tượng**. Hoàng Kiếm. Nhà xuất bản Giáo dục. 1997
3. **Giáo trình Lập trình Hướng đối tượng nâng cao**. Hoàng Kiếm. Nhà xuất bản Giáo dục. 1998
4. **Giáo trình Lập trình Hướng đối tượng**. Tô Văn Nam. Nhà xuất bản Giáo dục. 2009
5. **Bài tập Lập trình Hướng đối tượng**. Tô Văn Nam. Nhà xuất bản Giáo dục. 2009
6. **Ngôn ngữ C và lập trình hướng đối tượng**. Bùi Thế Tâm. Nhà xuất bản Giao thông vận tải. 2006
7. **Robocode Home,** URL: http://robocode.sourceforge.net/, last visited 08/08/2016

PHỤ LỤC 1
LẬP TRÌNH VỚI ROBOCODE

Giới thiệu về Robocode

Robocode là một thư viện lập trình cho phép lập trình viên có thể tạo ra các xe tăng để đấu nhau với các xe tăng của các lập trình viên khác.

Để tải về Robocode ta vào http://robocode.sourceforge.net/ và làm theo hướng dẫn.

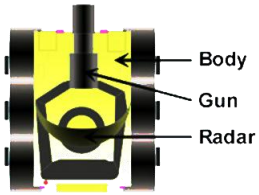
Thêm thư viện Robocode vào NetBeans

Ta *chọn New Project* như bình thường, sau đó vào *Properties* rồi chọn *Add JAR*. Tiếp theo đó tìm đến thư mục **libs** của Robocode và chọn **robocode.jar**

Để kết hợp với Javadocs của Robocode, ấn vào *Edit* và chọn thư mục Javadoc của Robocode

Đối với Eclipse làm tương tự.

Cấu trúc của một Robot



Robot có 3 phần: Thân, súng và radar. Ba phần này có thể di chuyển độc lập. Thân có thể quay được và di chuyển tiến hoặc lùi và chuyển động chậm nhất. Súng có thể quay được và chuyển động nhanh hơn thân. Radar cũng có thể quay được và chuyển động nhanh nhất.

Robot sử dụng một nguồn năng lượng chung cho cả tấn công và phòng thủ. Khi bị bắn trúng hay khi robot đâm vào tường, năng lượng của robot sẽ bị giảm. Súng sẽ không bắt được nếu bị overheat (gọi đến phương thức getGunHeat() để xác định giá trị này)

Viết mã nguồn cho Robot

Chọn *New Class* như bình thường. Lớp này kế thừa từ **robocode.Robot** hoặc có thể `import robocode.*` và kế thừa lớp **Robot**.

Sau đó ta có thể ghi đè (override) lại các phương thức của lớp **Robot**.

Chạy thử robot trong Robocode

Chọn menu *Options* | *Preferences*, vào tab *Development* rồi add thư mục đến gói chứa các package của mình (thư mục **classes** trong NetBeans hoặc thư mục **bin** trong Eclipse).

Một số phương thức quan trọng

- **`setColors`** - đổi màu robot
- **`setAdjustRadarForGunTurn(true);`** - cho phép camera chuyển động độc lập với súng
- **`getHeading()`** - trả về góc hiện tại của robot
- **`getBearing()`** - phương thức của event (ví dụ khi scan được một enemy thì đối tượng đó là một event), trả về góc của robot so với đối tượng đó.
- **`getDistance()`** - trả về khoảng cách từ robot đến đối tượng

AdvancedRobot

Các robot thông thường sẽ thực hiện ngay khi lệnh được gọi. Để có thể kết hợp nhiều việc cùng lúc (ví dụ: quét và bắt) ta có thể viết các robot nâng cao bằng cách khai báo lớp kế thừa AdvancedRobot

Khi chúng ta viết mã nguồn cho các lớp robot nâng cao, thay vì việc gọi trực tiếp đến fire, ahead, turnRight… ta phải gọi đến các phương thức set* tương ứng. Khi chúng ta muốn robot thực hiện tất cả các yêu cầu đó cùng lúc, ta bắt buộc phải gọi đến phương thức **`execute()`**