**BÀI TẬP MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

**(OJBECT ORIENTED PROGRAMMING EXERCISES)**

**HỆ: ĐẠI HỌC**

***Bài 1.*** Viết chương trình tính diện tích, chu vi hình chữ nhật.

* Hãy viết lớp **HinhChuNhat** gồm có:
  + Attributes : chiều dài, chiều rộng.
  + Phương thức thiết lập (set), và lấy (get) thông tin chiều dài, chiều rộng.
  + Phương thức tính diện tích, chu vi.
  + Phương thức toString gồm các thông tin dài, rộng, diện tích, chu vi.
* Xây dựng lớp chứa hàm main cho phần kiểm nghiệm. Dài rộng có thể nhập từ bàn phím.

***Bài 2.*** Viết chương trình OOP quản lý sinh viên đơn giản: Nhập, xuất thông tin, tính điểm TB.

* Viết lớp Sinh viên như sau: Attributes (private):
  + Mã sinh viên là số nguyên.  Họ tên: chuỗi ký tự.
  + Điểm LT, điểm TH : float Constructor:
  + Constructor mặc định (để khởi tạo đối tượng với các thông tin kiểu số là 0, kiểu chuỗi là chuỗi rỗng).
  + Constructor thứ hai nhận đầy đủ thông tin để khởi tạo giá trị cho tất cả các biến instance.

Methods:

* + Các getter và setter cho mỗi thuộc tính.
  + Tính điểm trung bình.
  + Phương thức toString để diễn tả đối tượng ở dạng chuỗi.
* Xây dựng class chứa hàm main: tạo 3 đối tượng sinh viên sv1, sv2, sv3, trong đó:
  + sv1 chứa thông tin của chính mình (tạo bằng constructor đủ thông số, thông tin biết rồi khỏi nhập từ bàn phím).
  + sv2 là thông tin người bạn thân nhất của em (tạo bằng constructor đủ thông số, thông tin biết rồi khỏi nhập từ bàn phím).
  + sv3 tạo bằng constructor mặc định. Nhập các thông tin cho sv3 từ bàn phím rồi sau đó dùng các setter để gán vào cho các thuộc tính tương ứng.
  + In bảng danh sách sinh viên gồm 4 cột là MSSV, họ tên, điểm LT, điểm TH, điểm TB (bảng có 3 dòng cho 3 sinh viên).

HD: phương thức xuất của đối tượng sinh viên in thông tin trên một dòng có định dạng. Sử dụng

System.out.printf(“chuỗi định dạng”, đối số 1, đối số 2, ….); Trong đó chuỗi định dạng giống c++, ví dụ:

“%-30s”: chuỗi, chiếm 30 ký tự, dấu trừ canh lề trái.

“%5.2f” : số thực, chiếm 5 ký tự, bao gồm 2 ký số lẻ.

Ký tự định dạng:

* + s : chuỗi
  + d: số nguyên (byte, short, int, long)
  + f: số thực (float, double)  b: boolean

***Bài 3.*** Sở giao thông cần theo dõi việc đăng ký xe của người dân. Dựa vào thông tin trị giá xe và dung tích xylanhcủa xe, sở giao thông cũng tính mức thuế phải đóng trước bạ khi mua xe như sau:

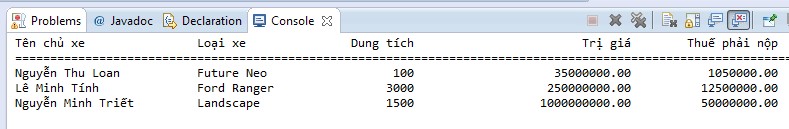
* Dưới 100cc, 1% trị giá xe.
* Từ 100 đến 200cc, 3% trị giá xe.
* Trên 200cc, 5% trị giá xe.

Hãy thiết kế và cài đặt class Vehicle với các attributes và methods phù hợp. Class phải có các constructor và phải bảo đảm tính encapsulation.

Xây dựng class chứa hàm main. Hàm main in ra menu lựa chọn các công việc:

* 1. Nhập thông tin và tạo các đối tượng xe1, xe2, xe3
  2. Xuất bảng kê khai tiền thuế trước bạ của các xe.
  3. Thoát.

Mẫu kết xuất của chương trình:



***Bài 4.*** Ngân hàng ABC muốn lưu trữ thông tin của mỗi tài khoản như sau: Mỗi tài khoản chứa các thông tin:

* Số tài khoản ( Kiểu long),
* Tên tài khoản (kiểu chuỗi),
* Số tiền trong tài khoản (kiểu double)

(a). Thiết kế lớp Account để lưu trữ các thông tin, lớp bao gồm các phương thức sau:

* Constructor: Có 2 constructor ( mặc định và đầy đủ tham số)
* Các phương thức get, set cho từng thuộc tính
* Phương thức toString để trả về chuỗi chứa toàn bộ thông tin tài khoản, yêu cầu định dạng tiền tệ.

(b). Thêm các thông tin sau vào lớp Account

* Hằng số lãi suất có giá trị khởi tạo 0.035
* Constructor có 2 đối số: số tài khoản, tên tài khoản. Constructor này sẽ khởi tạo số tiền mặc định là 50.
* Phương thức nạp tiền vào tài khoản: Lấy số tiền hiện tại trong tài khoản + số tiền nạp vào
* Phương thức rút tiền: Lấy số tiền hiện tại trong tài khoản – (số tiền muốn rút+phí rút tiền)
* Phương thức đáo hạn: Mỗi lần đến kỳ đáo hạn thì số tiền trong tài khoản = số tiền trong tài khoản + số tiền trong tài khoản \* LAISUAT
* Phương thức chuyển khoản từ tài khoản này sang tài khoản khác

Chú ý: Mỗi thao tác phải kiểm tra số tiền nạp, rút, chuyển có hợp lệ hay không? (VD: tiền

nhập vào <0, tiền rút nhiều hơn tiền trong tài khoản thì thông báo không hợp lệ và yêu cầu nhập lại)

***Bài 5.*** Một phân số gồm có 2 thuộc tính là tử số (numerator) và mẫu số (denominator). Hãy thiết kế class Phân số (Rational) để thực hiện các chức năng sau:

* + Tối giản phân số (reduce)
  + Nghịch đảo phân số (reciprocal)
  + Cộng 2 phân số (add)
  + Trừ 2 phân số (subtract)
  + Nhân 2 phân số (multiply)
  + Chia 2 phân số (divide)
  + So sánh 2 phân số, sử dụng sai số 0.0001(tolerance) để so sánh.

***Bài 6.*** Viết lớp **HangThucPham** mô tả một hàng hóa là hàng thực phẩm trong kho của một siêu thị, có các thuộc tính: mã hàng *(không cho phép sửa, không được để rỗng)*, tên hàng *(không được để rỗng)*, đơn giá *(>0)*, ngày sản xuất và ngày hết hạn *(ngày không được để rỗng, ngày hết hạn phải sau ngày sản xuất)*.

Ràng buộc chặt chẽ các ràng buộc trên các trường dữ liệu. Nếu dữ liệu không hợp lệ thì gán giá trị mặc định cho phép tương ứng của trường đó.

* + Tạo 1 constructor có đầy đủ tham số, 1 constructor có tham số là mã hàng.
  + Viết các phương thức setters/getters.
  + Viết phương thức kiểm tra một hàng thực phẩm đã hết hạn chưa?
  + Phương thức toString, trả về chuỗi chứa thông tin của hàng thực phẩm. *Trong đó: Định dạng đơn giá có phân cách hàng nghìn. Định dạng kiểu ngày là dd/MM/yyyy.*

Viết lớp cho phần kiểm nghiệm.

***Bài 7.*** a. Viết chương trình xây dựng đối tượng Sinh viên gồm có các thuộc tính sau: o Mã sinh viên là số nguyên, o Họ tên: chuỗi ký tự, o Địa chỉ: chuỗi ký tự, o Số điện thoại: là số bao gồm 7 chữ số

* + Các thuộc tính khai báo private, định nghĩa các phương thức get/set cho từng thuộc tính.
  + Viết các constructor để khởi tạo đối tượng (constructor mặc định, constructor có tham số).
  + Override phương thức toString của lớp Object.

b. Xây dựng lớp cho phép nhập vào N sinh viên *(dùng mảng)*. Xuất danh sách trên ra màn hình theo thứ tự tăng dần của mã sinh viên.

***Bài 8.*** a. Viết chương trình xây dựng đối tượng CD gồm có các thuộc tính sau:

o Mã CD là số nguyên, o Tựa CD: chuỗi ký tự, o Ca sỹ: chuỗi ký tự, o Số bài hát: số nguyên (>0) o Giá thành : số thực (>0)

* + Các thuộc tính khai báo private, định nghĩa các phương thức get/set cho từng thuộc tính.
  + Viết các constructor để khởi tạo đối tượng CD.
  + Override phương thức toString của lớp Object.

b. Xây dựng lớp lưu danh sách các CD *(dùng mảng)*.

* + Phương thức thêm 1 CD vào danh sách, thêm thành công nếu không trùng mã CD và kích thước mảng còn cho phép.
  + Tính số lượng CD có trong danh sách.
  + Tình tổng giá thành của các CD.
  + Phương thức sắp xếp danh sách giảm dần theo giá thành.
  + Phương thức sắp xếp danh sách tăng dần theo tựa CD.
  + Phương thức xuất toàn bộ danh sách.

c. Viết lớp cho phần kiểm nghiệm. Dùng menu case thực hiện các chức năng theo yêu cầu.

**Tuần 3.**

**Tuần 4. CÁC KHÁI NIỆM CƠ BẢN KẾ THỪA VÀ ĐA HÌNH**

**Chƣơng 4. Kế thừa và đa hình trên Java** *Mục tiêu:*

* *Hiểu và áp dụng được cách viết code kế thừa và đa hình trên Java.*
* *Hiểu và áp dụng được cách sử dụng mô hình lớp với mô tả kế thừa.*

*Yêu cầu:*

* *Tất cả các bài tập lưu trong thư mục: D:\MaSV\_HoTen\Tuan03-04\*
* *Máy tính phải được cài đặt sẵn JDK (Java Development Kit)*
* *Máy tính phải có sẵn phần mềm soạn thảo hỗ trợ cho lập trình hướng đối tượng dùng ngôn ngữ lập trình Java (Eclipse/JCreator/NetBeans)*  *Tên tập tin .java phải cùng tên với tên lớp.*

***Bài 1.*** Công ty du lịch V quản lý thông tin là các chuyến xe. Thông tin của 2 loại chuyến xe:

* Chuyến xe nội thành: Mã số chuyến, Họ tên tài xế, số xe, số tuyến, số km đi được, doanh thu.
* Chuyến xe ngoại thành: Mã số chuyến, Họ tên tài xế, số xe, nơi đến, số ngày đi được, doanh thu. Thực hiện các yêu cầu sau:
* Xây dựng các lớp với chức năng thừa kế.
* Viết chương trình quản lý các chuyến xe theo dạng cây thừa kế với các phương thức sau:
  + Nhập, xuất danh sách các chuyến xe (danh sách có thể dùng cấu trúc mảng).
  + Tính tổng doanh thu cho từng loại xe.

*Hướng dẫn:*

B1. Xây dựng lớp Chuyến xe bao gồm các thuộc tính chung cho cả chuyến xe ngoại thành và chuyến xe nội thành: *mã chuyến xe, tên tài xế, số xe, doanh thu*.

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** ChuyenXe  {  **protected** String Macx, NameTX, SoX; **protected** **double** DoanhThu;  Scanner in = **new** Scanner(System.*in*);    **protected** **void** setMacx(String macx) {  Macx = macx;  }    **protected** String getMacx() { **return** Macx;  } |

|  |  |
| --- | --- |
| **protected** **void** setNameTX(String nameTX) {  NameTX = nameTX;  }    **protected** String getNameTX() { **return** NameTX;  }    **protected** **void** setSoX(String soX) {  SoX = soX;  }    **protected** String getSoX() {  **return** SoX;  }    **protected** **void** setDoanhThu(**double** doanhThu) {  DoanhThu = doanhThu;  }    **protected** **double** getDoanhThu() {  **return** DoanhThu;  }    **protected** ChuyenXe()  {  **this**.Macx=""; **this**.NameTX=""; **this**.SoX="";  **this**.DoanhThu=0;  }    **protected** ChuyenXe(String ma, String name, String sx, **double** dt)  {  **this**.Macx=ma; **this**.NameTX=name; **this**.SoX=sx;  **this**.DoanhThu=dt;  }    **protected** **void** nhap()  {  System.*out*.println("Nhap MaSo Chuyen: "); **this**.Macx=in.nextLine();  System.*out*.println("Ho Ten Tai Xe: "); **this**.NameTX=in.nextLine(); System.*out*.println("So Xe: "); **this**.SoX=in.nextLine();  System.*out*.println("Doanh Thu: "); **this**.DoanhThu=in.nextDouble();  }    **public** String toString()  { | |
| }  } | **return** "\nMS chuyen: " + **this**.Macx + "\nName Tai Xe: " + **this**.NameTX + "\nSo Xe: " + **this**.SoX; |

B2. Xây dựng lớp Chuyến xe Ngoại thành thừa kế lớp Chuyến xe bao gồm thuộc tính: nơi đến, số ngày.

|  |
| --- |
| **import** java.util.Scanner;  **class** NgoaiThanh **extends** ChuyenXe  { **private** String NoiDen; **private** **int** SoNgay;  Scanner in = **new** Scanner(System.*in*);    **public** **void** setNoiDen(String noiDen) {  NoiDen = noiDen;  }  **public** String getNoiDen() { **return** NoiDen;  }  **public** **void** setSoNgay(**int** soNgay) {  SoNgay = soNgay;  }  **public** **int** getSoNgay() { **return** SoNgay;  }    **public** NgoaiThanh()  { **super**(); **this**.SoNgay=0;  **this**.NoiDen="";  }    **public** NgoaiThanh(String ma, String name, String sx, **double** dt, String nd, **int** sn) {  **super**(ma, name, sx, dt); **this**.SoNgay=sn; **this**.NoiDen=nd;  }    **public** **void** nhap()  {  **super**.nhap();  System.*out*.println("Noi Den: "); **this**.NoiDen=in.nextLine(); System.*out*.println("So Ngay: "); **this**.SoNgay=in.nextInt();  }    **public** String toString() |

{

**return** "Chuyen Xe Ngoai Thanh" + **super**.toString()

+ "\nNoi Den: " + **this**.NoiDen + "\nSo Ngay: " + **this**.SoNgay

+ "\nDoanh Thu: " + **this**.DoanhThu;

}

}

B3. Xây dựng lớp Chuyến xe Nội thành thừa kế lớp Chuyến xe bao gồm thuộc tính: số km, số tuyến.

|  |
| --- |
| **import** java.util.Scanner; **class** NoiThanh **extends** ChuyenXe  { **private** **double** SoKm; **private** String SoTuyen;  Scanner in = **new** Scanner(System.*in*);    **public** NoiThanh()  { **super**(); **this**.SoTuyen=""; **this**.SoKm=0;  }    **public** NoiThanh(String ma, String name, String sx, **double** dt, String st, **double** skm)  {  **super**(ma, name, sx, dt); **this**.SoTuyen=st; **this**.SoKm=skm;  }    **public** **void** setSoKm(**double** soKm) {  SoKm = soKm;  }    **public** **double** getSoKm() { **return** SoKm;  }  **public** **void** setSoTuyen(String soTuyen) {  SoTuyen = soTuyen;  }    **public** String getSoTuyen() { **return** SoTuyen;  }    **public** **void** nhap()  {  **super**.nhap();  System.*out*.println("Nhap So Tuyen: "); **this**.SoTuyen=in.nextLine(); System.*out*.println("So Km: "); **this**.SoKm=in.nextDouble(); |

}

**public** String toString()

{

**return** "Chuyen Xe Noi Thanh " + **super**.toString()

+ "\n So Tuyen: " + **this**.SoTuyen + "\nSo Km: " + **this**.SoKm

+ "\nDoanh Thu: " + **this**.DoanhThu;

}

}

B4: Xây dựng lớp quản lý danh sách các chuyến xe (dùng cấu trúc mảng)

|  |  |
| --- | --- |
| **public** **class** ListChuyenXe  {  **private** ChuyenXe[] CX = **new** ChuyenXe[100]; **private** **int** countCX;  **private** **double** sum1=0, sum2=0;    **public** **void** ListCX()  { countCX=0;  **for**(**int** i=0; i<100; i++)  {  CX[i] = **new** ChuyenXe();  }  }    **public** **void** listcx(**int** temp)  {  **if**(countCX>100)  System.*out*.println("Khong the them du lieu!");  **else**  {  **if**(temp==1)  {  CX[countCX] = **new** NoiThanh(); NoiThanh noit = **new** NoiThanh();  noit.nhap(); CX[countCX]=noit;  sum1+=noit.getDoanhThu();  }  **else**  {  CX[countCX] = **new** NgoaiThanh(); NgoaiThanh ngoait = **new** NgoaiThanh();  ngoait.nhap(); CX[countCX]=ngoait;  sum2+=ngoait.getDoanhThu();  }  countCX++;  }  }    **public** **void** xuattt() | |
| {                    }  } | **for**(**int** i=0 ; i<countCX; i++)  {  System.*out*.println("------------/---------------/------------"); System.*out*.println(CX[i].toString());  }  System.*out*.println("----------------Doanh Thu-----------------");  System.*out*.println("| Chuyen Xe Noi Thanh: " + sum1 + "|");  System.*out*.println("| Chuyen Xe Ngoai Thanh: " + sum2 + "|");  System.*out*.println("------------------------------------------"); |

B5. Xây dựng lớp quản lý thông tin cho phép nhập xuất và tính doanh thu cho từng xe.

|  |
| --- |
| **import** java.util.Scanner; **public** **class** QLChuyenXe  {  **public** **static** **void** main(String[] args)  { **int** x;  Scanner a = **new** Scanner(System.*in*);    ListChuyenXe dscx = **new** ListChuyenXe();  **do**  {  System.*out*.println("\n\t-----------------------------------");  System.*out*.println("\t0. Exit");  System.*out*.println("\t1. Them Chuyen Xe Noi Thanh");  System.*out*.println("\t2. Them Chuyen Xe Ngoai Thanh");  System.*out*.println("\t3. Hien thi danh muc Chuyen Xe"); System.*out*.println("\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"); x = a.nextInt();  System.*out*.println("------>>"); **switch**(x)  { **case** 1:  dscx.listcx(1); **break**; **case** 2:  dscx.listcx(2); **break**; **case** 3: dscx.xuattt(); **break**;  }  } **while**(x!=0);  System.*out*.println(""); }  } |

***Bài 2.*** Thư viên X quản lý danh sách các loại sách . Thông tin về các loại sách:

* Sách giáo khoa: Mã sách, ngày nhập (ngày, tháng, năm), đơn giá, số lượng, nhà xuất bản, tình trạng (mới, cũ).

Nếu tình trạng sách là mới thì: thành tiền = số lượng \* đơn giá.

Nếu tình trạng sách là cũ thì: thành tiền = số lượng \* đơn giá \* 50%

* Sách tham khảo: Mã sách, ngày nhập (ngày, tháng, năm), đơn giá, số lượng, nhà xuất bản, thuế.

Thành tiền = số lượng \* đơn giá +thuế Thực hiện các yêu cầu sau:

* Xây dựng các lớp với chức năng thừa kế.
* Nhập xuất danh sách các loại sách.
* Tính tổng thành tiền cho từng loại.
* Tính trung bình cộng đơn giá của các sách tham khảo.
* Xuất ra các sách giáo khoa của nhà xuất bản X.

***Bài 3.*** Xây dựng chương trình quản lý danh sách các giao dịch. Hệ thống quản lý 2 loại giao dịch:

* Giao dịch vàng: Mã giao dịch, ngày giao dịch (ngày, tháng, năm), đơn giá, số lượng, loại vàng.

**import**

java.util.Scanner;

**public**

**class**

GiaoDich

{

**protected**

**int**

MaGD

,

SoLuong

;

Thành tiền được tính như sau:

thành tiền = số lượng \* đơn giá.

* Giao dịch tiền tệ: Mã giao dịch, ngày giao dịch (ngày, tháng, năm), Đơn giá, số lượng, tỉ giá, loại tiền tệ có 3 loại: tiền Việt Nam, tiền USD, tiền Euro. Thành tiền được tính như sau:

- Nếu là tiền USD hoặc Euro thì: thành tiền = số lượng \* đơn giá\* tỉ giá - Nếu là tiền VN thì: thành tiền = số lượng \* đơn giá Thực hiện các yêu cầu sau:

* Xây dựng các lớp với chức năng thừa kế.
* Nhập xuất danh sách các giao dịch.
* Tính tổng số lượng cho từng loại.
* Tính trung bình thành tiền của giao dịch tiền tệ.
* Xuất ra các giao dịch có đơn giá > 1 tỷ.

*Hướng dẫn:*

B1. Xây dựng lớp Giao dịch bao gồm các thuộc tính chung cho cả giao dịch vàng và giao dịch ngoại tệ: *mã giao dịch, số lượng, ngày giao dịch, đơn giá, thành tiền*.

|  |
| --- |
| **protected** String NgayGD; **protected** **double** DonGia, ThanhTien;  Scanner in = **new** Scanner(System.*in*);    **protected** GiaoDich()  { **this**.MaGD=0; **this**.NgayGD=""; **this**.DonGia=0; **this**.SoLuong=0;  **this**.ThanhTien=0;  }    **protected** GiaoDich(**int** ma, **int** sl, String ngay, **double** dgia, **double** ttien)  {  **this**.MaGD=ma; **this**.NgayGD=ngay; **this**.DonGia=dgia; **this**.SoLuong=sl;  **this**.ThanhTien=ttien;  }    **protected** **int** getMaGD() { **return** MaGD;  }  **protected** **void** setMaGD(**int** maGD) {  MaGD = maGD;  }  **protected** **int** getSoLuong() { **return** SoLuong;  }  **protected** **void** setSoLuong(**int** soLuong) {  SoLuong = soLuong;  }  **protected** String getNgayGD() { **return** NgayGD;  }  **protected** **void** setNgayGD(String ngayGD) {  NgayGD = ngayGD;  }  **protected** **double** getDonGia() { **return** DonGia;  }  **protected** **void** setDonGia(**double** donGia) {  DonGia = donGia;  }  **protected** **double** getThanhTien() {  **return** ThanhTien;  }    **protected** **void** setThanhTien(**double** thanhTien) {  ThanhTien = thanhTien;  }    **protected** **void** nhap() |
| {  System.*out*.println("Ma Giao Dich: ");  MaGD = in.nextInt();  System.*out*.println("Ngay Giao Dich: ");  NgayGD = in.next();  System.*out*.println("Don Gia: ");  DonGia = in.nextDouble();  System.*out*.println("So Luong: ");  SoLuong = in.nextInt();  }    **public** String toString()  {  **return** " [MaGD=" + MaGD + ", SoLuong=" + SoLuong + ", NgayGD="  + NgayGD + ", DonGia=" + DonGia; }  } |

B2. Xây dựng lớp Giao dịch vàng thừa kế lớp Giao dịch bao gồm thuộc tính: *loại vàng*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **class** GiaoDichVang **extends** GiaoDich  {  **private** String LoaiVang;  **public** GiaoDichVang()  { **super**(); **this**.LoaiVang=""; | | | |
|  | | }    **publi**  String l  { | | **c** GiaoDichVang(**int** ma, **int** sl, String ngay, **double** dgia, **double** ttien,  vang)  **super**(ma, sl, ngay, dgia, ttien); **this**.LoaiVang=lvang; | |
|  | |  | }  **public** String getLoaiVang() {  **return** LoaiVang;  }  **public** **void** setLoaiVang(String loaiVang) {  LoaiVang = loaiVang;  }  **public** **void** nhap()  {  **super**.nhap();  System.*out*.println("Loai Vang: ");  LoaiVang = in.next(); | | |
|  | |  | } **publi** { | **c** **double** TTien() | |
| **return** **this**.ThanhTien=**this**.DonGia\***this**.SoLuong;  }    **public** String toString()  {  **return** "Giao Dich Vang: " + **super**.toString() + ", LoaiVang: " + LoaiVang  + ", ThanhTien: " + ThanhTien + "]"; }  } | | | |

B3. Xây dựng lớp Ngoại tệ thừa kế lớp Giao dịch bao gồm thuộc tính: *tỷ giá và loại tiền tệ (USD, Euro, Việt Nam)*.

|  |
| --- |
| **class** GiaoDichTienTe **extends** GiaoDich  { **private** **float** TiGia;  **private** **int** LoaiTienTe;    **public** GiaoDichTienTe()  { **super**(); **this**.TiGia=0;  **this**.LoaiTienTe=0;  }    **public** GiaoDichTienTe(**int** ma, **int** sl, String ngay, **double** dgia, **double** ttien, **float** tgia, **int** loai)  {  **super**(ma, sl, ngay, dgia, ttien);  **this**.TiGia=tgia;  **this**.LoaiTienTe=loai;  }    **public** **float** getTiGia() { **return** TiGia;  }  **public** **void** setTiGia(**float** tiGia) {  TiGia = tiGia;  }  **public** **int** getLoaiTienTe() {  **return** LoaiTienTe;  }  **public** **void** setLoaiTienTe(**int** loaiTienTe) {  LoaiTienTe = loaiTienTe;  }    **public** **void** nhap()  {  **super**.nhap();  System.*out*.println("Ti Gia: ");  TiGia=in.nextFloat();  System.*out*.println("Loai Tien Te: "); |

|  |
| --- |
| LoaiTienTe=in.nextInt();  }    **public** **double** TTien()  {  **if**(LoaiTienTe==1)  **return** **this**.ThanhTien=**this**.DonGia\***this**.SoLuong;  **else**  **return** **this**.ThanhTien=**this**.DonGia\***this**.SoLuong\***this**.TiGia;  }    **public** String toString()  {  String temp; **if**(LoaiTienTe==1) temp="VND";  **else** **if**(LoaiTienTe==2)  temp="USD"; **else** temp="EURO";  **return** "Giao Dich Tien Te: " + **super**.toString() + "' Ti Gia: " + TiGia  + ", Loai Tien Te: " + temp + ", ThanhTien: " + ThanhTien + "]";  }  } |

B4: Xây dựng lớp quản lý danh sách các giao dịch (dùng cấu trúc mảng)

|  |
| --- |
| **public** **class** ListGD  {  GiaoDich gd[] = **new** GiaoDich[100];  **private** **int** CountGD, sum1=0, sum2=0, dem=0, sumtien=0; **public** ListGD()  {  CountGD=0;  **for**(**int** i=0; i<100; i++) gd[i] = **new** GiaoDich();  }  **public** **void** ThemGD(**int** temp)  {  **if**(CountGD>100)  System.*out*.println("Bo nho full, ko the them moi!"); **else**  {  **if**(temp==1)  {  gd[CountGD] = **new** GiaoDichVang(); GiaoDichVang gdv = **new** GiaoDichVang();  gdv.nhap();  gdv.ThanhTien=gdv.TTien(); gd[CountGD] = gdv; sum1+=gdv.SoLuong;  }  **else** |
| {  gd[CountGD] = **new** GiaoDichTienTe(); GiaoDichTienTe gdtt = **new** GiaoDichTienTe();  gdtt.nhap();  gdtt.ThanhTien=gdtt.TTien(); gd[CountGD] = gdtt; sumtien+=gdtt.ThanhTien;  sum2+=gdtt.SoLuong;  dem++;  }  CountGD++;  }  }  **public** **void** HienThiDS()  {  **for**(**int** i=0; i<CountGD; i++)  {  System.*out*.println("\n----So TT: " + (i+1));  System.*out*.println(gd[i].toString());  }  }  // Tong so luong cua giao dich **public** **void** TongSL()  {  System.*out*.println("|-> Tong So Luong cua Giao Dich Vang: " + sum1 + " <-|");  System.*out*.println("|-> Tong So Luong cua Giao Dich Tien Te:" + sum2 + " <-|");  }    **public** **void** TrungBinhGDTT()  {  System.*out*.println("|-> Trung Binh Thanh Tien Giao Dich Tien Te: "  + (sumtien/dem) + " <-|");  }  // xuat danh sach giao dich 1 ty **public** **void** GiaoDich1ty()  { **int** d=0;  **for**(**int** i=0; i<CountGD; i++)  {  **if**(gd[i].DonGia>1000)  {  System.*out*.println(gd[i].toString());  d++;  }  }  **if**(d==0)  System.*out*.println("Khong co Thong Tin!");  }  } |

B5. Xây dựng lớp quản lý thông tin cho giao dịch.

**import** java.util.Scanner;

|  |
| --- |
| **public** **class** QLGD  {  **public** **static** **void** main(String[] args)  {  ListGD lgd =**new** ListGD();  Scanner in = **new** Scanner(System.*in*);  **int** x; **do**  {  System.*out*.println("\n\t\t----------------------------------");  System.*out*.println("\t\t|0. Thoat ung dung\t\t\t\t |");  System.*out*.println("\t\t|1. Them Giao Dich Vang\t\t\t\t |");  System.*out*.println("\t\t|2. Them Giao Dich Tien Te\t\t\t |");  System.*out*.println("\t\t|3. Hien Thi Danh Sach Giao Dich\t\t |");  System.*out*.println("\t\t|4. Xem Tong So Luong cua cac Giao Dich\t\t |");  System.*out*.println("\t\t|5. Xem Trung Binh Cua Giao Dich Tien Te\t |");  System.*out*.println("\t\t|6. Hien Thi Danh Sach Giao Dich co Don Gia > 1 ngan|");  System.*out*.println("\t\t--------------------------------------"); x=in.nextInt(); **switch**(x)  { **case** 1:  lgd.ThemGD(1); **break**; **case** 2:  lgd.ThemGD(2); **break**; **case** 3:  lgd.HienThiDS(); **break**; **case** 4: lgd.TongSL(); **break**; **case** 5:  lgd.TrungBinhGDTT();  **break**; **case** 6:  lgd.GiaoDich1ty();  **break**;  }  }**while**(x!=0);  System.*out*.println("");  }  } |

***Bài 4.*** Xây dựng chương trình quản lý danh sách các giao dịch nhà đất. Thông tin bao gồm:

* Giao dịch đất: Mã giao dịch, ngày giao dịch (ngày, tháng, năm), đơn giá, loại đất (loại A, B, C), diện tích.
  + Nếu là loại B, C thì: thành tiền = diện tích \* đơn giá.
  + Nếu là loại A thì: thành tiền = diện tích \* đơn giá \* 1.5
* Giao dịch nhà: Mã giao dịch, ngày giao dịch (ngày, tháng, năm), đơn giá, loại nhà (cao cấp, thường), địa chỉ, diện tích.
  + Nếu là loại nhà cao cấp thì: thành tiền = diện tích \* đơn giá.
  + Nếu là loại thường thì: thành tiền = diện tích \* đơn giá \* 90% Thực hiện các yêu cầu sau:
* Xây dựng các lớp với chức năng thừa kế.
* Nhập xuất danh sách các giao dịch.
* Tính tổng số lượng cho từng loại.
* Tính trung bình thành tiền của giao dịch đất.
* Xuất ra các giao dịch của tháng 9 năm 2013.

***Bài 5.*** Xây dựng chương trình quản lý danh sách hoá đơn tiền điện của khách hàng. Thông tin bao gồm các loại khách hàng :

* Khách hàng Việt Nam: mã khách hàng, họ tên, ngày ra hoá đơn (ngày, tháng, năm), đối tượng khách hàng (sinh hoạt, kinh doanh, sản xuất): số lượng (số KW tiêu thụ), đơn giá, định mức.

Thành tiền được tính như sau:

* + Nếu số lượng <= định mức thì: thành tiền = số lượng \* đơn giá.
  + Ngược lại thì: thành tiền = số lượng \* đơn giá \* định mức + số lượng KW vượt định mức \* Đơn giá \* 2.5.
* Khách hàng nước ngoài: mã khách hàng, họ tên, ngày ra hoá đơn (ngày, tháng, năm), quốc tịch, số lượng, đơn giá. Thành tiền được tính = số lượng \* đơn giá. Thực hiện các yêu cầu sau:
* Xây dựng các lớp với chức năng thừa kế.  Nhập xuất danh sách các hóa đơn khách hàng.
* Tính tổng số lượng cho từng loại khách hàng.
* Tính trung bình thành tiền của khách hàng người nước ngoài.
* Xuất ra các hoá đơn trong tháng 09 năm 2013 (cùa cả 2 loại khách hàng).

*Hướng dẫn:*

B1. Xây dựng lớp Khách hàng bao gồm các thuộc tính chung cho cả Khách hàng nước ngoài và Khách hàng Việt Nam. Gồm các thuộc tính: *mã khách hàng, số lượng, đơn giá, thành tiền, ngày của hóa đơn và họ tên khách hàng.*

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** KhachHang  {  **protected** **int** MaKH, SoLuong; **protected** **double** DonGia, ThanhTien; **protected** String NgayHD, HoTenKH;  Scanner in =**new** Scanner(System.*in*);    **protected** KhachHang()  { **this**.MaKH=0; **this**.DonGia=0; **this**.NgayHD=""; **this**.HoTenKH="";  **this**.SoLuong=0;  }    **protected** KhachHang(**int** ma, **double** dgia, String ngay, String hoten, **int** sl, **double** ttien)  {  **this**.MaKH=ma; **this**.DonGia=dgia; **this**.NgayHD=ngay; **this**.HoTenKH=hoten; **this**.SoLuong=sl;  **this**.ThanhTien=ttien;  }    **protected** **int** getMaKH() { **return** MaKH;  }  **protected** **void** setMaKH(**int** maKH) {  MaKH = maKH;  }  **protected** **double** getDonGia() { **return** DonGia;  }  **protected** **void** setDonGia(**double** donGia) {  DonGia = donGia;  }  **protected** String getNgayHD() { **return** NgayHD;  }  **protected** **void** setNgayHD(String ngayHD) {  NgayHD = ngayHD;  } |
| **protected** String getHoTenKH() { **return** HoTenKH;  }    **protected** **void** setHoTenKH(String hoTenKH) {  HoTenKH = hoTenKH;  }  **protected** **int** getSoLuong() { **return** SoLuong;  }    **protected** **void** setSoLuong(**int** sl) {  SoLuong = sl;  }  **protected** **double** getThanhTien() {  **return** ThanhTien;  }    **protected** **void** setThanhTien(**double** thanhTien) {  ThanhTien = thanhTien;  }    **protected** **void** nhap()  {  System.*out*.println("Nhap Ma Khach Hang: ");  MaKH = in.nextInt();  System.*out*.println("Ho va Ten Khach Hang: ");  HoTenKH = in.next();  System.*out*.println("Ngay Lap Hoa Don: ");  NgayHD = in.next();  System.*out*.println("So Luong(KW): ");  SoLuong = in.nextInt();  System.*out*.println("Don Gia: ");  DonGia = in.nextDouble();  }    **public** String toString() { **return** "[MaKH: " + MaKH + ", HoTenKH: " + HoTenKH + ", NgayHD: " + NgayHD  + ", DonGia: " + DonGia + ", SoLuong: " +SoLuong;  }  } |

B2. Xây dựng lớp Khách hàng nước ngoài thừa kế lớp Khách hàng bao gồm thuộc tính: *quốc tịch.*

|  |
| --- |
| **class** KhachHangNuocNgoai **extends** KhachHang  { **private** String QuocTich;    **public** KhachHangNuocNgoai()  { **super**();  **this**.QuocTich="";  } |
| **public** KhachHangNuocNgoai(**int** ma, **double** dgia, String ngay, String hoten, **int** sl, **double** ttien, String qt)  {  **super**(ma, dgia, ngay, hoten, sl, ttien);  **this**.QuocTich=qt;  }    **public** String getQuocTich() {  **return** QuocTich;  }    **public** **void** setQuocTich(String quocTich) {  QuocTich = quocTich;  }    **public** **void** nhap()  {  **super**.nhap();  System.*out*.println("Quoc Tich: ");  QuocTich = in.next();  }    **public** **double** TTien()  {  **return** **this**.ThanhTien = **this**.SoLuong\***this**.DonGia;  }    **public** String toString()  {  **return** "Khach Hang Nuoc Ngoai: " + **super**.toString() + ", QuocTich: " + QuocTich  + ", ThanhTien: " + ThanhTien; }  } |

B3. Xây dựng lớp Khách hàng Việt Nam thừa kế lớp Khách hàng bao gồm thuộc tính: *loại khách hàng, định mức.*

|  |
| --- |
| **class** KhachHangViet **extends** KhachHang  {  **private** **int** LoaiKH, DinhMuc;    **public** KhachHangViet()  { **super**(); **this**.LoaiKH=0;  **this**.DinhMuc=0;  }    **public** KhachHangViet(**int** ma, **double** dgia, String ngay, String hoten, **int** sl, **double** ttien, **int** loai, **int** dm)  {  **super**(ma, dgia, ngay, hoten, sl, ttien); |
| **this**.LoaiKH = loai;  **this**.DinhMuc = dm;  }  **public** **int** getLoaiKH() { **return** LoaiKH;  }  **public** **void** setLoaiKH(**int** loaiKH) {  LoaiKH = loaiKH;  }  **public** **int** getDinhMuc() { **return** DinhMuc;  }    **public** **void** setDinhMuc(**int** dinhMuc) {  DinhMuc = dinhMuc;  }    **public** **void** nhap()  {  **super**.nhap();  System.*out*.println("Loai Khach Hang (1=SinhHoat, 2=KinhDoanh, 3=SanXuat)");  LoaiKH = in.nextInt();  System.*out*.println("Dinh Muc: ");  DinhMuc = in.nextInt();  }    **public** **double** TTien()  {  **if**(SoLuong<=DinhMuc)  **return** **this**.ThanhTien=**this**.DonGia\***this**.SoLuong;  **else**  **return** **this**.ThanhTien=**this**.DinhMuc\***this**.DonGia + (**this**.SoLuong**this**.DinhMuc)\***this**.DonGia\*2.5;  }    **public** String toString()  {  String temp; **if**(LoaiKH==1) temp="Sinh Hoat"; **else** **if**(LoaiKH==2)  temp="Kinh Doanh"; **else**  temp="San Xuat";  **return** "Khach Hang Viet: " + **super**.toString() + ", LoaiKH: " + temp  + ", Dinh Muc: " + DinhMuc + ", Thanh Tien: " + ThanhTien; }  } |

B4: Xây dựng lớp quản lý danh sách các khách hàng (dùng cấu trúc mảng)

**public** **class** KhachHangList

{

|  |
| --- |
| **private** KhachHang kh[] = **new** KhachHang[100]; **private** **int** CountKH, sum1=0, sum2=0, dem=0, sumtien=0;    **public** KhachHangList()  {  CountKH=0;  **for**(**int** i=0; i<100; i++) kh[i] = **new** KhachHang();  }    **public** **void** ThemKH(**int** temp)  {  **if**(CountKH>100)  System.*out*.println("Bo Nho Da Day, ko the them!"); **else**  {  **if**(temp==1)  {  kh[CountKH] = **new** KhachHangViet(); KhachHangViet khv = **new** KhachHangViet();  khv.nhap();  khv.ThanhTien = khv.TTien(); kh[CountKH]=khv;  sum1+=khv.SoLuong;  }  **else**  {  kh[CountKH] = **new** KhachHangNuocNgoai();  KhachHangNuocNgoai khnn = **new** KhachHangNuocNgoai();  khnn.nhap();  khnn.ThanhTien = khnn.TTien(); kh[CountKH] = khnn; sum2 += khnn.SoLuong;  sumtien += khnn.ThanhTien;  dem++;  }  CountKH++;  }  }    **public** **void** HienThiKH()  {  **for**(**int** i=0; i<CountKH; i++)  {  System.*out*.println("\nSo TT: " + (i+1));  System.*out*.println(kh[i].toString());  }  }    **public** **void** TongSL()  {  System.*out*.println(":::->> Tong So Luong(KW) Dien cua KH Viet Nam: " + sum1);  System.*out*.println(":::->> Tong So Luong(KW) Dien cua KH Nuoc Ngoai: "  + sum2); |

}

**public** **void** TrungBinh()

{

System.*out*.println("|-->> Trung Binh Thanh Tien cua KH Nuoc Ngoai: " + (sumtien/dem));

}

}

B5. Xây dựng lớp quản lý thông tin cho phép nhập xuất và tính trung bình thành tiền.

|  |
| --- |
| **import** java.util.Scanner; **public** **class** KhachHang\_CTQuanLy  {  **public** **static** **void** main(String[] args)  {  KhachHangList lkh = **new** KhachHangList(); Scanner in = **new** Scanner(System.*in*);  **int** x;    **do**  {  System.*out*.println("\n\t\t------------------------------------");  System.*out*.println("\t\t|0. Thoat ung dung\t\t\t\t |");  System.*out*.println("\t\t|1. Them Khach Hang Viet\t\t\t |");  System.*out*.println("\t\t|2. Them Khach Hang Nuoc Ngoai\t\t\t |");  System.*out*.println("\t\t|3. Hien Thi Danh Sach Khach Hang\t\t |");  System.*out*.println("\t\t|4. Xem Tong (KW)cua cac KH\t\t |");  System.*out*.println("\t\t|5. Xem TB Thanh Tien cua KH NN\t |"); System.*out*.println("\t\t--------------------------------------"); x=in.nextInt();  System.*out*.println("======>>"); **switch**(x)  { **case** 1: lkh.ThemKH(1); **break**; **case** 2: lkh.ThemKH(2); **break**; **case** 3: lkh.HienThiKH(); **break**; **case** 4: lkh.TongSL(); **break**; **case** 5: lkh.TrungBinh(); **break**;  }  }**while**(x!=0);  System.*out*.println("");  }  } |

***Bài 6.*** Một khách sạn X cần quản lý các hóa đơn của khách hàng thuê phòng. Hóa đơn có 2 loại: hóa đơn theo giờ, hóa đơn theo ngày . Thông tin chung của chi tiết hóa đơn là: Mã hóa đơn, ngày hóa đơn (ngày, tháng, năm), Tên khách hàng, mã phòng, đơn giá. Thông tin riêng của từng loại hóa đơn gồm:

* Hóa đơn theo giờ còn có số giờ thuê. Thành tiền = số giờ thuê \* đơn giá. Nếu trường hợp số giờ > 24 tiếng và < 30 tiếng thì cũng chỉ tính 24 giờ. Nếu trường hợp số giờ là > 30 tiếng thì không dùng loại hóa đơn theo giờ.
* Hóa đơn theo ngày sẽ có số ngày thuê. Thành tiền = số ngày thuê \* đơn giá. Nếu số ngày >7 thì giảm 20% đơn giá cho những ngày còn lại. Thực hiện các yêu cầu sau:
* Xây dựng các lớp với chức năng thừa kế.
* Nhập xuất danh sách các hóa đơn thuê phòng.  Tính tổng số lượng cho từng loại thuê phòng.
* Tính trung bình thành tiền của hóa đơn thuê phòng trong tháng 9/2013.

***Bài 7.*** *(Yêu cầu chung: mặc định access modifier (private, protected, public) cho từng thuộc tính/ phương thức mỗi lớp, cài đặt getter/setter, cài đặt constructor mặc định và constructor có thông số đầy đủ).*

Giả sử cần xây dựng chương trình quản lý dùng cho một học viện nghiên cứu giảng dạy và ứng dụng. Đối tượng quản lý bao gồm các sinh viên đang theo học, các nhân viên đang làm việc tại học viện, các khách hàng đến giao dịch mua bán sản phẩm ứng dụng. Dựa vào một số đặt tính của từng đối tượng, người quản lý cần đưa ra cách thức đánh giá khác nhau.

Vậy hãy xây dựng các lớp sau:

* *Lớp* ***Person****: bao gồm các thuộc tính họ tên, địa chỉ, phương thức toString.*
* Các *lớp* ***Student, Employee, Customer*** (mô tả dưới đây) thừa kế lớp ***Person***.
  + Lớp ***Student:*** bao gồm các thuộc tính điểm môn học 1, điểm môn học 2, và các phương thức: tính điểm TB, đánh giá, overriding phương thức toString trả về bảng điểm sinh viên (gồm thông tin thuộc tính và điểm TB).
  + Lớp ***Employee:*** bao gồm thuộc tính *heSoLương,*  và các phương thức: tính lương, đánh giá, overriding phương thức toString trả về bảng lương cho nhân viên (gồm thông tin thuộc tính đối tượng và tiền lương). o Lớp ***Customer:*** bao gồm thuộc tính tên công ty, trị giá hóa đơn, đánh giá, và phương thức toString trả về thông tin hóa đơn cho khách hàng (gồm các thuộc tính của đối tượng).
* Lớp có 1 *biến* danh sách để lưu các sinh viên, nhân viên, khách hàng (dùng 1 biến array Person), biến lưu tổng số người có trong danh sách, constructor mặc định khởi tạo array với dung lượng cho trước, phương thức thêm một người vào danh sách (thông số Person), xóa 1 người khỏi danh sách (nhận thông số là họ tên của người cần xóa), sắp xếp danh sách theo thứ tự họ tên, phương thức xuất danh sách. Khi danh sách đầy thì tự động tăng dung lượng dãy lên 50%.
* Viết lớp với phương thức main cho phần kiểm nghiệm. Giao tiếp với người dùng bằng menu *(thể hiện tính đa hình – polymorphism bằng cách cho phép lựa chọn nhập thông tin là sinh viên, nhân viên hay khách hàng).*

***Bài 8.*** Hàng hóa quản lý trong kho của một siêu thị gồm có hàng thực phẩm, hàng sành sứ và hàng điện máy.

Mỗi loại hàng đều có mã hàng *(không được sửa, không được để trống)*, tên hàng *(không được rỗng)*, số lượng tồn *(>=0)*, đơn giá *(>0)*.

Hàng thực phẩm thì cần quan tâm đến thông tin ngày sản xuất, ngày hết hạn *(ngày hết hạn phải sau hoặc là ngày sản xuất)* và nhà cung cấp.

Hàng điện máy cần biết thời gian bảo hành bao nhiêu tháng *(>=0)*, công suất bao nhiêu KW *(>0)*.

Hàng sành sứ thì cần biết thông tin về nhà sản xuất và ngày nhập kho.

Ngoài ra, người quản lý cần quan tâm đến số lượng tồn kho và các yếu tố khác của từng loại hàng hóa để đánh giá mức độ bán buôn, tiền VAT từng loại hàng hóa. *Biết rằng VAT của hàng điện máy và sành sứ là 10%, VAT của hàng thực phẩm là 5%.*

a) Dựa vào các thông tin trên, hãy xác định:

* Các lớp có thể có. Lớp nào là lớp trừu tượng *(abstract class)*, lớp nào là lớp cụ thể .
* Các thuộc tính cho từng lớp.
* Các phương thức cho từng lớp (phương thức nào là phương thức trừu tượng *(abstract method),* danh sách các tham số có thể có cho từng phương thức và kiểu trả về của phương thức).
* Thiết kế mô hình class *(xây dựng cây thừa kế, các giao diện nếu có)*.

b) Dùng java IDE, tạo một project. Thực hiện cài đặt tường minh cho mỗi loại hàng cụ thể trên. Trong đó, để đánh giá mức độ bán buôn thì:

* Hàng điện máy, nếu số lượng tồn kho <3 thì được đánh giá là bán được.
* Hàng thực phẩm, nếu vẫn còn tồn kho và bị hết hạn thì đánh giá là khó bán. o Hàng sành sứ, nếu số lượng tồn kho >50 và thời gian lưu kho >10 ngày thì đánh giá là bán chậm.
* Các trường hợp còn lại xem như không đánh giá.
* Hãy viết lớp quản lý danh sách hàng hóa. Dùng Array để lưu trữ danh sách hàng hóa.
  + Tạo constructor khởi tạo danh sách. o Viết phương thức thêm một hàng hóa vào danh sách *(thêm thành công nếu không bị trùng mã hàng, thể hiện tính đa hình – polymorphism bằng cách cho phép lựa chọn nhập thông tin)* o Viết phương thức in toàn bộ danh sách các hàng hóa.
* Tạo lớp cho phần thử nghiệm, với menu lựa chọn để thực hiện các chức năng theo yêu cầu.

-

***Bài 9.*** Với 1 tập mini các loại xe trong thế giới thực cho bên dưới:



**Yêu cầu quản lý:**

* Thông tin từng loại xe.
* Tính tiền thuế cho từng chiếc xe dựa trên giá trị xe như sau:
  + **Xe đạp:** Không đóng thuế. o **Xe máy**: VAT=10% và thuế trước bạ 5%. o **Xe ô tô khách**: Thuế tiêu thụ đặc biệt 30% (số chỗ >=5); 50% (số chỗ<5), thuế VAT=10%, thuế trước bạ 20%.
  + **Xe ô tô tải:** VAT=10%, thuế trước bạ 2%. **Yêu cầu sinh viên:**
* Dùng kiến thức mô hình hóa dữ liệu trong lập trình hướng đối tượng để xây dựng các lớp.
* Sử dụng 1 CASE tool *(Computer Aided Software Engineering)* để thiết kế mô hình cây phân cấp các lớp. Mối quan hệ giữa các lớp và các interface.
* Generate source code từ mô hình trên sang ngôn ngữ lập trình Java.
* Hiện thực tường minh chương trình.

***Bài 10.*** Lớp ***Thô g ti co gười***: bao gồm các thuộc tính *tên, địa chỉ* và các phương thức *nhập tên, nhâp địa chỉ.*

* Các lớp ***Sinh viên, Nhân viên, Khách hàng*** (mô tả dưới đây) thừa kế lớp ***Thô g ti co gười***.
* Lớp ***Sinh viên:*** bao gồm các thuộc tính *điểm môn học 1, điểm môn học 2,*  và các phương thức *nhập điểm, tính tổng điểm, phương thức thay đổi thông tin của sinh viên truyền vào tham số là sinh viên (truyền tham biến ) , phương thức cho phép lựa chọn cách thay đổi – 0: thay đổi điểm môn 1, -1 thay đổi điểm môn 2, điểm các môn được nhập từ bàn phím.*
* Lớp ***Nhân viên:*** bao gồm các thuộc tính *lương, chức vụ,*  và các phương thức *nhập chức vụ, tính lương, phương thức thay đổi thông tin của nhân viên truyền vào tham số là nhân viên, phương thức cho phép lựa chọn cách thay đổi – 0: thay đổi lương, -1 thay đổi chức vụ, các thông tin được nhập từ bàn phím.*
* Lớp ***Khách hàng:*** bao gồm các thuộc tính *kiểu xe bán,*  và các phương thức *nhập kiểu xe, xuất hóa đơn.*
* Hãy định nghĩa các lớp trên và lớp ***Hiển thị thông tin.*** Lớp ***Hiển thị thông tin*** cho phép khai báo đối tượng thuộc lớp ***Thô g ti co gười,*** thể hiện tính đa hình (polymorphism) bằng cách cho phép lựa chọn nhập thông tin là sinh viên, nhân viên hay khách hàng.

***Bài 11.*** Để quản lý thông tin về sinh viên của trường ĐHCN, thông tin về sinh viên được tổ chức như sau:

* Sinh viên trung cấp (trung cấp 2 năm)
* Sinh viên cao đẳng
* Sinh viên đại học
* Sinh viên liên kết (Austraulia, US)

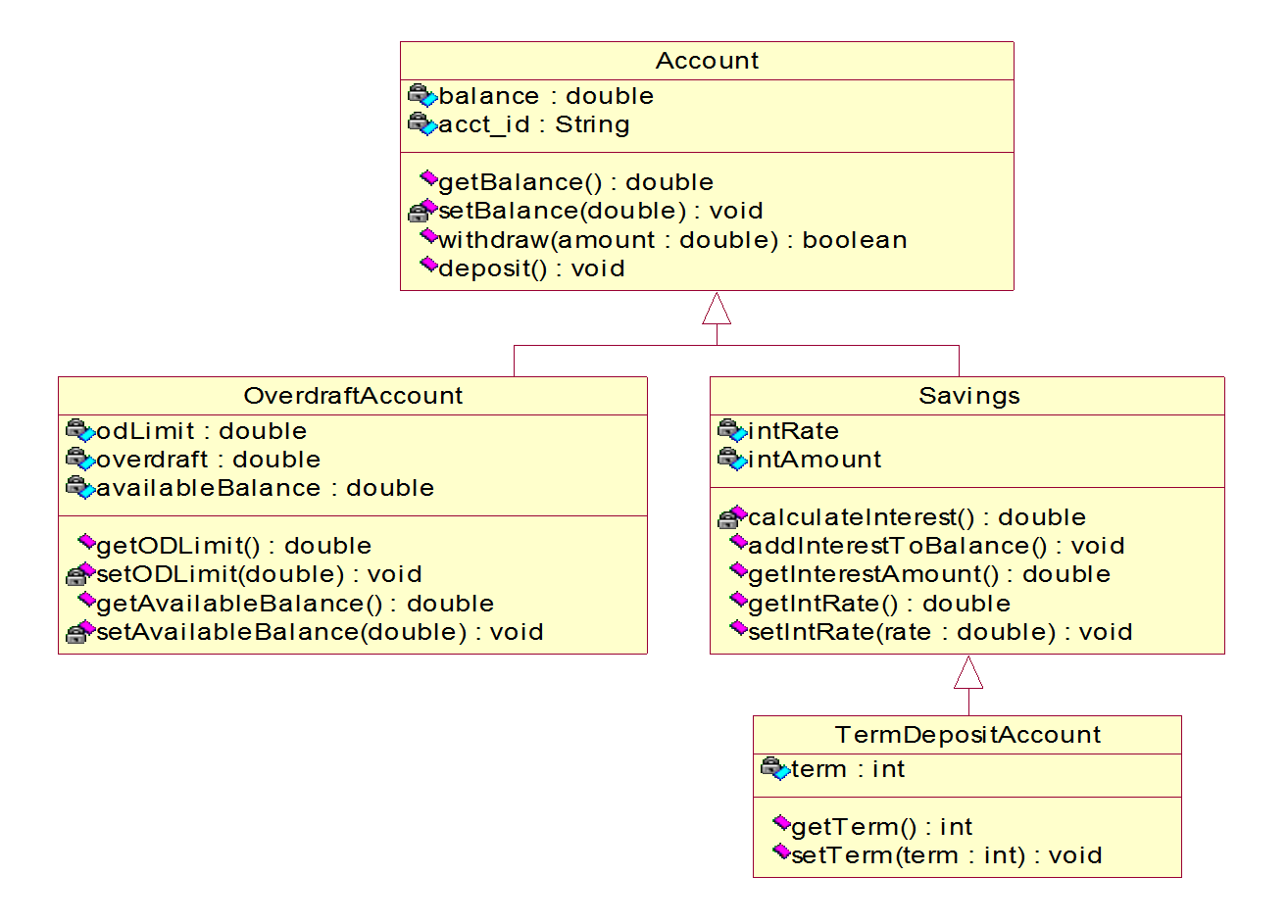
1. Xác định các thông tin, dữ liệu chung, riêng.
2. Định nghĩa các lớp (các thuộc tính, và phương thức) và mô hình phân cấp các lớp
3. Viết lớp cho phép nhập và hiển thị thông tin về sinh viên.

***Bài 12.*** Tạo lớp trừu tượng (*abstract* class) **Shape** với 3 phương thức trừu tượng **draw()** , **erase( )**, and **move(int** *x***, int** *y* **)**. Tạo các lớp con như liệt kê ở bảng dưới đây đồng thời ghi đè (overriding) các phương thức trừu tượng, các phương thức này in câu thông báo ra console.

|  |  |  |
| --- | --- | --- |
| **Class** | **Superclass** | **Subclass** |
| Shape | - | Circle, Quad, Triangle, Polygon |
| Circle | Shape | - |
| Quad | Shape | Rectangle |
| Rectangle | Quad | - |
| Triangle | Shape | - |
| Polygon | Shape | - |

Viết lớp **Drawing** có phương thức **drawShape(Shape** *theShape***)**, phương thức có tham số là đối tượng **Shape**. Trong phương thức gọi tới **draw() của từng đối tƣợng** Shape.

***Bài 13.*** Cho mô hình như sau



1. Sử dụng mô hình lớp ở trên cho cây phân cấp của các đối tượng Account, hãy tạo các phương thức để thi hành phần thiết kế này. Cung cấp các đoạn mã lệnh để thực thi các phương thức.
2. Viết chương trình kiểm tra trong lớp BankingServices, tạo các đối tượng thuộc các lớp cụ thể và kiểm tra các phương thức.

**Tuần 5.**

**Tuần 6. TẬP HỢP (COLLECTIONS) VÀ MẢNG**

**Chƣơng 5. Tập hợp và mảng trên Java** *Mục tiêu:*

|  |  |
| --- | --- |
|  | *Hiểu và áp dụng được cách thao tác trên mảng* |
|  | *Hiểu và áp dụng được cách thao tác trên tập hợp*   * *Khai báo và khởi tạo tập hợp* * *Các thao tác thêm, xóa, sửa, duyệt trên tập hợp* * *Các thuật toán sắp xếp, tìm kiếm trên tập hợp* |

*Yêu cầu:*

|  |  |
| --- | --- |
|  | *Tất cả các bài tập lưu trong thư mục: D:\MaSV\_HoTen\Tuan05-06\* |
|  | *Máy tính phải được cài đặt sẵn JDK (Java Development Kit)* |
|  | *Máy tính phải có sẵn phần mềm soạn thảo hỗ trợ cho lập trình hướng đối tượng dùng ngôn ngữ lập trình Java (Eclipse/JCreator/NetBeans)* |
|  | *Tên tập tin .java phải cùng tên với tên lớp.* |

***Bài 1.*** Công ty TrueLove:

1. Công ty TrueLove cần lưu tên của các nhân viên của mình. Mỗi tháng một nhân viên sẽ được chọn ngẫu nhiên để nhận một quà tặng. Hãy dùng tuyển tập để viết chương trình quản lý danh sách nhân viên.
2. Công ty TrueLove cần đặt tên cho sản phẩm mới, tên sản phẩm được chọn từ tên của nhân viên, vì vậy tên không được trùng, tên chỉ được dùng có 1 lần. Hãy dùng tuyển tập để viết chương trình cung cấp tên cho sản phẩm.
3. Công ty TrueLove muốn dùng tên phổ biến nhất cho sản phẩm của họ, tên phổ biến là tên giống nhau nhiều nhất. Hãy dùng tuyển tập để viết chương trình cung cấp tên cho sản phẩm.
4. Công ty TrueLove muốn cho nhân viên đi du lịch, chính sách được tạo ra là ưu tiên cho những người đăng ký trước. Hãy dùng tuyển tập để viết chương trình đăng ký du lịch.
5. Công ty TrueLove muốn tạo danh sách các khách hàng theo thứ tự tăng dần theo doanh số. Hãy dùng tuyển tập để viết chương trình quản lý danh sách khách hàng.

***Bài 2.*** Phòng học được quản lý trong một trường đại học gồm: phòng học lý thuyết, phòng máy tính và phòng thí nghiệm .

Mỗi phòng học đều có mã phòng, dãy nhà, diện tích, số bóng đèn.

Phòng học lý thuyết thì cần quan tâm xem có máy chiếu không.

Phòng máy tính thì cần biết là trang bị bao nhiêu máy tính.

Phòng thí nghiệm thì thêm thông tin chuyên ngành, sức chứa, có bồn rửa không *(rửa dụng cụ thí nghiệm / rửa tay).*

Ngoài ra, người quản lý cần phải xem xét phòng học có đạt chuẩn không.

**Dùng java IDE, tạo một project đƣợc đặt tên theo quy định sau: TênLớp\_TênSV\_MSSV.**

* Thực hiện cài đặt tường minh cho mỗi loại phòng cụ thể trên.
* Phòng học đạt chuẩn nếu: Tất cả các phòng đều phải đủ ánh sáng *(trung bình 10m2 - 1 bóng đèn)*, vào Phòng lý thuyết, nếu có máy chiếu. o Phòng máy tính, nếu trung bình 1.5m2 đặt một máy.
  + Phòng thí nghiệm, nếu có bồn rửa đi kèm.
* Hãy viết lớp quản lý danh sách phòng học. Dùng một List (ArrayList, LinkedList, Vector) để lưu trữ danh sách phòng học.
  + Tạo constructor khởi tạo danh sách. o Viết phương thức thêm một phòng học vào danh sách *(thêm thành công nếu không bị trùng mã phòng)*.
  + Viết phương thức tìm kiếm một phòng học nào đó khi biết mã phòng. o Viết phương thức in toàn bộ danh sách các phòng học. o Viết các phương thức để in danh sách các phòng học đạt chuẩn. o Viết phương thức để sắp xếp danh sách tăng dần theo cột dãy nhà. o Viết phương thức để sắp xếp danh sách giảm dần theo cột diện tích. o Viết phương thức để sắp xếp danh sách tăng dần theo cột số bóng đèn. o Viết phương thức để cập nhật số máy tính cho một phòng máy tính nào đó khi biết mã phòng. o Viết phương thức để xóa một phòng học nào đó khi biết mã phòng. *Lưu ý khi test chương trình, khi xóa cần phải xác minh rằng có chắc chắn xóa không?* o Viết phương thức để in ra tổng số phòng học.
  + Viết các phương thức để in danh sách các phòng máy có 60 máy.
* Tạo lớp cho phần thử nghiệm, với menu lựa chọn để thực hiện các chức năng theo yêu cầu.

***Bài 3.***  Quản lý khách hàng xếp hàng mua vé tại nhà ga. Thông tin lưu trữ cho khách hàng gồm: số CMND khác hàng (String), Tên khách hàng, Ga đến, giá tiền (double).

Hệ thống menu gồm các mục:

* + Thêm một khách hàng mới vào hàng đợi mua vé. o Bán một vé cho khách hàng. Chỉ bán cho người đăng ký trước. o Hiển thị danh sách khách hàng. o Hủy một khách hàng ra khỏi danh sách. (khách hàng không mua vé nữa).
  + Thống kê tình hình bán vé o Lưu danh sách vào file o Hiển thị danh sách các ga đang chờ mua vé.
  + Hiển thị danh sách các ga đang chờ mua vé và số vé tương ứng cho ga.

Lưu ý:

* + Số khách hàng trong danh sách hiện tại là số khách đang chờ, nhưng chưa có vé. Khi một khách hàng đã mua vé, thì loại khách hàng này ra khỏi danh sách chờ mua vé.
  + Việc mua vé phải có thứ tự: ai vào trước thì mua vé trước (FIFO).
  + Mỗi khi khách hàng mua được vé phải lưu lại khách hàng này để dùng cho việc thống kê. o Mỗi khi thêm một khác hàng mới, nếu Số CMND khách hàng đã có thì không tạo phần tử mới mà chỉ cập nhật lại ga và giá tiền đến cho khác hàng đó.
  + Mục thống kê tình hình: cho biết còn bao nhiêu khách hàng chờ nhận vé, bao nhiêu khách hàng đã nhận vé, tổng số tiền đã thu về là bao nhiêu. o Việc lưu danh sách: chỉ lưu các khách hàng chờ mua vé. Các khách hàng đã nhận vé xem như kết sổ trong ngày không cần lưu lại.
  + Khi chương trình vừa được chạy, lập tức tự động nạp toàn bộ danh sách khách hàng từ file (cách khách hàng chưa có vé).
  + Khi hiển thị danh sách các ga đến đang chờ mua vé, chỉ hiển thị tên ga đó một lần. (Ví dụ:

giả sử 10 khách hàng nhưng đăng ký đi đến 2 ga, thì chỉ hiển thị 2 hàng).

***Bài 4.*** Viết chương trình tạo 2 tập hợp các số nguyên (Set). Tính giao, hội, hiệu 2 tập trên, xuất ra kết quả tăng dần.

**Hƣớng dẫn:** - Dùng TreeSet

|  |  |  |  |
| --- | --- | --- | --- |
|  | a.addAll(b) |  |  tập a hội tập b |
|  | a. retainAll(b) |  | tập a giao tập b |

 a. removeAll(b) tập a trừ tập b

***Bài 5.*** Sử dụng ArrayList để biểu diễn một vài chức năng của interface Collection . Chương trình thực hiện sử dụng 2 mảng Color trong ArrayLists và dùng Iterator để loại bỏ các phần tử trong mảng thứ 2 của tập hợp ArrayList từ mảng thứ nhất của tập hợp ArrayList.

***Bài 6.*** Viết lớp mô tả các toán tử trên danh sách liên kết sử LinkedList. Chương trình tạo 2

LinkedList chứa thông tin là các chuỗi String. Các phần tử của danh sách List này được đưa vào danh sách kia. Các chuỗi trong danh sách được chuyển sang chữ hoa, xoá các phần tử.

***Bài 7.*** Viết chương trình tra cứu danh bạ điện thoại, sử dụng cấu trúc collection bất kỳ cho phù hợp để lưu trữ thông tin của danh bạ và dễ dàng thực hiện công việc:

* Tra cứu theo địa chỉ .
* Tra cứu theo số điện thoại, 1 địa chỉ có thể đăng kí nhiều số điện thoại cố định.

**Tuần 7.**

**Tuần 8. LẬP TRÌNH GENERICS**

**Chƣơng 5. Lập trình Generics** *Mục tiêu:*

* *Hiểu và áp dụng được các kiểu Generic, phương thức Generic*
* *Hiểu và áp dụng được Generic theo lớp*

*Yêu cầu:*

|  |
| --- |
| * *Tất cả các bài tập lưu trong thư mục: D:\MaSV\_HoTen\Tuan07-08\* * *Máy tính phải được cài đặt sẵn JDK (Java Development Kit)* * *Máy tính phải có sẵn phần mềm soạn thảo hỗ trợ cho lập trình hướng đối tượng dùng ngôn ngữ lập trình Java (Eclipse/JCreator/NetBeans)* * *Trong cài đặt Generic sử dụng*   *E - Element (used extensively by the Java Collections Framework)*  *K - Key*  *N - Number*  *T – Type, S,U,V etc. - 2nd, 3rd, 4th types*  *V - Value* |

***Bài 1.*** Viết 1 lớp chung, constructor cho phép khởi tạo dữ liệu bất kỳ (xét trường hợp vừa là số vừa là chuỗi).

*Hướng dẫn:*

Tạo lớp Generic

|  |
| --- |
| **class** GenericClass<T>  {  T ob;  GenericClass(T o)  { ob = o;  }  T getob() {  **return** ob;  }  **void** showType()  {  System.*out*.println("Kieu cua T la " + ob.getClass().getName()); }  } |

Tạo lớp kiểm tra

|  |
| --- |
| **public** **class** GenericClassTest  {  **public** **static** **void** main(String args[])  {  // Tao đối tượng cho GenericClass lưu trữ các số Integer. GenericClass<Integer> iOb = **new** GenericClass<Integer>(88); iOb.showType();    // Không cần ép kiểu **int** v = iOb.getob();  System.*out*.println("gia tri: " + v);    // Tao đối tượng cho GenericClass lưu trữ các chuỗi String.  GenericClass<String> strOb = **new** GenericClass<String>("Generics Test"); strOb.showType();    String str = strOb.getob();  System.*out*.println("gia tri: " + str);  } }  } |

|  |  |
| --- | --- |
| Generic Term | Meaning |
| Set<E> | Generic Type , E is called formal parameter.  Loại Generic, E được gọi là tham số hình thức. |
| Set<Integer> | Parametrized type , Integer is actual parameter here Tham số hóa loại, Integer là tham số thực. |
| <T extends Comparable> | Bounded type parameter |
| <T super Comparable> | Bounded type parameter |
| Set<?> | Unbounded wildcard |
| <? extends T> | Bounded wildcard type |
| <? Super T> | Bounded wildcards |
| Set | Raw type |
| <T extends Comparable<T>> | Recursive type bound |

***Bài 2.*** Tạo lớp Generic với 2 loại tham số.

*Hướng dẫn:*

Tạo lớp với 2 loại tham số

|  |
| --- |
| **class** TwoGenerics<T, V>  {  T ob1;  V ob2;  TwoGenerics(T o1, V o2)  { ob1 = o1; ob2 = o2; |
| }  **void** showTypes()  {  System.*out*.println("Loai cua T la "  + ob1.getClass().getName());  System.*out*.println("Loai cua V la "  + ob2.getClass().getName());  }  T getob1() {  **return** ob1;  }  V getob2() {  **return** ob2;  }  } |

Tạo lớp kiểm tra

**public** **class** TwoGenericsTest

{

**public** **static** **void** main(String args[])

{

TwoGenerics<Integer, String> tgObj =

**new** TwoGenerics<Integer, String>(88, "Generics"); tgObj.showTypes();

**int** v = tgObj.getob1();

System.*out*.println("gia tri: " + v);

String str = tgObj.getob2();

System.*out*.println("gia tri: " + str);

}

}

***Bài 3.*** Thao tác với Generic lồng nhau (nested generic type). Tạo một danh sách List lưu trữ danh sách các danh sách (chuỗi).

*Hướng dẫn:*

|  |
| --- |
| **import** java.util.ArrayList; **import** java.util.List;  **public** **class** ListOfLists  {  **public** **static** **void** main(String[] args)  {  List<String> listOfStrings = **new** ArrayList<String>(); listOfStrings.add("Hello again"); List<List<String>> listOfLists = **new** ArrayList<List<String>>(); listOfLists.add(listOfStrings); |

***Bài 4.*** Sử dụng interface Generic Comparable. Tạo một lớp Person bao gồm 2 thuộc tính họ và tên implements interface Comparable.

|  |
| --- |
| String s = listOfLists.get(0).get(0);  // tự thêm thông tin khác  System.*out*.println(s); // ket qua "Hello again"  }  } |

Tạo lớp Person implements Generic Comparable

|  |
| --- |
| **import** java.util.Arrays;  **class** Person **implements** Comparable<Person>  { **private** String firstName; **private** String surname; **public** Person(String firstName, String surname)  {  **this**.firstName = firstName;  **this**.surname = surname;  }  **public** String toString()  {  **return** firstName + " " + surname;  }  **public** **int** compareTo(Person person)  {  **int** result = surname.compareTo(person.surname);  **return** result == 0 ? firstName.compareTo(((Person) person).firstName) : result;  }  } |

Tạo lớp kiểm tra.

|  |
| --- |
| **public** **class** PersonTest  {  **public** **static** **void** main(String[] args)  {  Person[] authors = { **new** Person("D", "S"), **new** Person("J", "G"), **new** Person("T", "C"), **new** Person("C", "S"), **new** Person("P", "C"), |

**new** Person("B", "B") };

Arrays.*sort*(authors); // Sắp xếp sử dụng phương thức Comparable

System.*out*.println("\Sau khi sap xep:"); **for** (Person author : authors) {

|  |
| --- |
| System.*out*.println(author);  }    Person[] people = { **new** Person("C", "S"), **new** Person("N", "K"), **new** Person("T", "C"), **new** Person("C", "D") }; **int** index = 0;  System.*out*.println("\nTim kiem:");    **for** (Person person : people)  {  index = Arrays.*binarySearch*(authors, person); **if** (index >= 0)  {  System.*out*.println(person +  " tai vi tri index " + index);  } **else**  {  System.*out*.println(person +  " khong tim thay. Gia tri tra ve: " + index);  }  } }  } |

***Bài 5.*** Thiết kế 1 lớp hoạt động như một là thư viện cho các loại Media sau: sách, video và báo chí. Yêu cầu viết cả 2 cách: thông thường và Generic.

*Hướng dẫn:*

Cách viết thông thường

|  |
| --- |
| **import** java.util.List; **import** java.util.ArrayList;  **public** **class** Library  {  **private** List resources = **new** ArrayList(); **public** **void** addMedia(Media x)  {  resources.add(x);  }  **public** Media retrieveLast()  {  **int** size = resources.size(); **if** (size > 0)  {  **return** (Media)resources.get(size - 1);  }  **return** **null**; |
| }  } **interface** Media  {  } **interface** Book **extends** Media  {  } **interface** Video **extends** Media  {  } **interface** Newspaper **extends** Media  {  } |

Cách viết theo Generic

|  |
| --- |
| **public** **class** LibraryGeneric<E **extends** Media>  {  **private** List<E> resources = **new** ArrayList<E>(); **public** **void** addMedia(E x)  {  resources.add(x);  }  **public** E retrieveLast()  {  **int** size = resources.size(); **if** (size > 0)  {  **return** resources.get(size - 1);  }  **return** **null**;  }  } |

***Bài 6.*** Viết phương thức Generic tính max, min trong một tập hợp. Sử dụng cách viết có dùng wildcard.

*Hướng dẫn:*

**import** java.util.Arrays;

**import** java.util.Collection; **import** java.util.Comparator;

**class** ComparatorsEx

{

**public** **static** <T> T max(Collection<? **extends** T> coll,

Comparator<? **super** T> cmp)

|  |
| --- |
| {  T candidate = coll.iterator().next(); **for** (T elt : coll)  {  **if** (cmp.compare(candidate, elt) < 0)  {  candidate = elt;  } }  **return** candidate;  } **public** **static** <T **extends** Comparable<? **super** T>>  T max(Collection<? **extends** T> coll)  {  **return** *max*(coll, ComparatorsEx.<T> *naturalOrder*());  }  **public** **static** <T> T min(Collection<? **extends** T> coll,  Comparator<? **super** T> cmp)  {  **return** *max*(coll, *reverseOrder*(cmp));  } **public** **static** <T **extends** Comparable<? **super** T>>  T min(Collection<? **extends** T> coll)  {  **return** *max*(coll, ComparatorsEx.<T> *reverseOrder*());  }  **public** **static** <T **extends** Comparable<? **super** T>>  Comparator<T> naturalOrder()  {  **return** **new** Comparator<T>()  {  **public** **int** compare(T o1, T o2)  {  **return** o1.compareTo(o2);  }  };  }  **public** **static** <T> Comparator<T> reverseOrder(**final** Comparator<T> cmp)  {  **return** **new** Comparator<T>() { **public** **int** compare(T o1, T o2)  {  **return** cmp.compare(o2, o1);  }  };  }  **public** **static** <T **extends** Comparable<? **super** T>>  Comparator<T> reverseOrder() |

{

|  |
| --- |
| **return** **new** Comparator<T>() { **public** **int** compare(T o1, T o2)  {  **return** o2.compareTo(o1);  }  };  }  } **public** **class** ComparatorsExampleTest  {  **public** **static** **void** main(String[] args)  {  Comparator<String> sizeOrder = **new** Comparator<String>() { **public** **int** compare(String s1, String s2) { **return** s1.length() < s2.length() ? -1 : s1.length() > s2.length() ? 1 :  s1.compareTo(s2);  }  };  Collection<String> strings = Arrays.*asList*("AAA", "aaa", "CCC", "f");  System.*out*.println(ComparatorsEx.*max*(strings));  System.*out*.println(ComparatorsEx.*min*(strings));  System.*out*.println(ComparatorsEx.*max*(strings,sizeOrder)); System.*out*.println(ComparatorsEx.*min*(strings,sizeOrder));  }  } |

***Bài 7.*** Viết phương thức Generic cho phép in ra mảng các phần tử, phương thức này cho phép in phần tử mảng của nhiều kiểu dữ liệu khác nhau.

*Hướng dẫn:*

|  |
| --- |
| **public** **class** PrintArrayGeneric  {  // phương thức Generic  **public** **static** <E> **void** printArray(E[] inputArray)  {  // hiển thị các phần tử mảng **for** (E element : inputArray)  System.out.printf("%s ", element);    System.out.println();  }    **public** **static** **void** main(String args[]) |

{

// 3 mảng dữ liệu khác nhau: Integer, Double, Character

Integer[] integerArray = { 1, 2, 3, 4, 5, 6 };

Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 }; Character[] characterArray = { 'H', 'E', 'L', 'L', 'O' };

System.out.println("Mang so nguyen:");

printArray(integerArray); // tham số là mảng số nguyên System.out.println("\nMang doubleArray co noi dung:"); printArray(doubleArray); // tham số là mảng Double System.out.println("\nMang characterArray co noi dung:"); printArray(characterArray); // tham số là mảng ký tự

}

}

***Bài 8.*** Sử dụng Generic tham số, dùng ký tự đại diện wildcard ?. Viết phương thức Generic cho phép in ra mảng các phần tử, phương thức sử dụng tham số Generic.

*Hướng dẫn:*

|  |
| --- |
| **import** java.util.ArrayList; **import** java.util.List;  **public** **class** GenericParameter  {  **public** **static** **void** printList (List<?> list)  {  **for** (Object element : list)  {  System.*out*.println(element);  } }  **public** **static** **void** main(String[] args)  {  List<String> list1 = **new** ArrayList<String>();  list1.add ("Hello"); list1.add ("World"); *printList* (list1);    List<Integer> list2 = **new** ArrayList<Integer>(); list2.add(100); list2.add(200); *printList*(list2);  }  } |

***Bài 9.*** Sử dụng bounded wildcard trong phương thức. Viết phương thức Generic cho phép tính trung bình các giá trị trong mảng.

*Hướng dẫn:*

Cú pháp: GenericType<? **extends** upperBoundType>

**import** java.util.ArrayList; **import** java.util.List;

**public** **class** BoundedWildcard

|  |
| --- |
| {  **public** **static** **double** getAverage(List<? **extends** Number> numberList)  {  **double** total = 0.0;  **for** (Number number : numberList)  {  total += number.doubleValue();  }  **return** total / numberList.size();  }  **public** **static** **void** main(String[] args)  {  List<Integer> integerList = **new** ArrayList<Integer>(); integerList.add(3); integerList.add(30); integerList.add(300);  System.*out*.println(*getAverage*(integerList)); // KQ?    List<Double> doubleList = **new** ArrayList<Double>(); doubleList.add(3.0); doubleList.add(33.0);  System.*out*.println(*getAverage*(doubleList)); // KQ?  }  } |

***Bài 10.*** Sử dụng bounded type trong lớp. Viết lớp Generic cho phép tính trung bình các giá trị trong mảng số (số nguyên/số thực).

*Hướng dẫn:*

|  |
| --- |
| **class** Stats<T **extends** Number>  {  T[] nums;  Stats(T[] o)  {  nums = o;  } **double** average()  {  **double** sum = 0.0;  **for**(**int** i=0; i < nums.length; i++) sum += nums[i].doubleValue();    **return** sum / nums.length; |

}

}

**public** **class** BoundedType

{

**public** **static** **void** main(String args[]) {

***Bài 11.*** Sử dụng bounded type trong lớp và wildcard ?. Viết lớp Generic cho phép tính trung bình các giá trị trong mảng và so sánh giá trị trung bình.

*Hướng dẫn:*

Viết lớp GenericStats dùng bounded type và phương thức so sánh 2 đối tượng thuộc lớp GenericStats dùng wildcard.

**class** GenericStats<T **extends** Number>

{

T[] nums;

GenericStats(T[] obj)

{

nums = obj;

}

|  |
| --- |
| Integer inums[] = { 1, 2, 3, 4, 5 };  Stats<Integer> iob = **new** Stats<Integer>(inums); **double** v = iob.average();  System.*out*.println("Trung binh iob: " + v);  Double dnums[] = { 1.1, 2.2, 3.3, 4.4, 5.5 }; Stats<Double> dob = **new** Stats<Double>(dnums); **double** w = dob.average();  System.*out*.println("Trung binh dob: " + w);  }  } |

**double** average()

{

**double** sum = 0.0;

**for**(**int** i=0; i < nums.length; i++)

{

sum += nums[i].doubleValue();

}

**return** sum / nums.length;

}

**boolean** sameAvg(GenericStats<?> ob)

{

**if**(average() == ob.average()) **return** **true**;

**return** **false**;

}

}

Viết lớp kiểm tra

**public** **class** GenericStatsTest {

|  |
| --- |
| **public** **static** **void** main(String args[])  {  Integer inums[] = { 1, 2, 3, 4, 5 };  GenericStats<Integer> iob = **new** GenericStats<Integer>(inums); **double** v = iob.average();  System.*out*.println("Trung binh cua iob " + v);    Double dnums[] = { 1.1, 2.2, 3.3, 4.4, 5.5 };  GenericStats<Double> dob = **new** GenericStats<Double>(dnums); **double** w = dob.average();  System.*out*.println("Trung binh cua dob " + w);    Float fnums[] = { 1.0F, 2.0F, 3.0F, 4.0F, 5.0F }; GenericStats<Float> fob = **new** GenericStats<Float>(fnums); **double** x = fob.average();  System.*out*.println("Trung binh cua fob " + x);    System.*out*.print("Trung binh cua iob va dob "); **if**(iob.sameAvg(dob))  {  System.*out*.println("giong nhau.");  } **else**  {  System.*out*.println("khac nhau.");  }    System.*out*.print("Trung binh cua iob va fob "); **if**(iob.sameAvg(fob))  {  System.*out*.println("giong nhau.");  } **else**  {  System.*out*.println("khac nhau.");  }  }  } |

***Bài 12.*** Viết phương thức override trong lớp Generic. Xét trường hợp lớp B thừa kế lớp A và viết override phương thức trong lớp A.

|  |
| --- |
| **class** Gen<T> {  T ob;  Gen(T o) { ob = o;  }    T getObject()  { |
| System.*out*.println("Gen's getObject(): " ); **return** ob;  }  } **class** Gen2<T> **extends** Gen<T>  {    Gen2(T o) { **super**(o);  }  T getObject()  {  System.*out*.println("Gen2's getObject(): "); **return** ob;  }  } **public** **class** OverrideGenericMethods  {  **public** **static** **void** main(String[] arg)  {  Gen<Integer> intObject = **new** Gen<Integer>(88);  Gen2<Long> longObject = **new** Gen2<Long>(99L);    intObject.getObject();  longObject.getObject();  } } |

Lưu ý: Thao tác ép kiểu trong Generic. Chỉ có thể ép kiểu một đối tượng thuộc lớp sang đối tượng của lớp khác trong trường hợp tương thích và tham số là như nhau.

|  |
| --- |
| **class** Gen<T> {  T ob;    Gen(T o) { ob = o;  }    T getObject()  { **return** ob;  }  }    **class** Gen2<T> **extends** Gen<T>  {  Gen2(T o) |

|  |
| --- |
| { **super**(o);  }  }    **public** **class** CastGeneric  {  **public** **static** **void** main(String args[])  {  Gen<Integer> intObject = **new** Gen<Integer>(88);  Gen2<Long> longObject = **new** Gen2<Long>(99L);    //longObject = (Gen2<Long>)intObject;    } } |

***Bài 13.*** Thao tác với Generic Interface. Viết Interface với 2 phương thức max, min để tính max, min trong một tập hợp. Lớp thực thi interface phải định nghĩa chi tiết việc xử lý của 2 phương thức này.

Cú pháp chung

1. type-param-list is a comma-separated list of type parameters.
2. When a generic interface is implemented, you must specify the type arguments

**interface interface**-name<type-param-list> { // ...

**class** class-name<type-param-list>

**implements** interface-name<type-param-list> {

**interface**

MinMax<T

**extends**

Comparable<T>>

{

T min();

T max();

}

**class**

MyClass<T

**extends**

Comparable<T>>

**implements**

MinMax<T>

{

T[]

vals

;

MyClass(T[] o)

{

vals

= o;

}

|  |
| --- |
| **public** T min()  {  T v = vals[0];  **for** (**int** i = 1; i < vals.length; i++) **if** (vals[i].compareTo(v) < 0) v = vals[i];    **return** v;  }    **public** T max()  {  T v = vals[0];  **for** (**int** i = 1; i < vals.length; i++) **if** (vals[i].compareTo(v) > 0) v = vals[i];    **return** v;  }  } **public** **class** GenericInterfaceTest  {  **public** **static** **void** main(String args[])  {  Integer inums[] = { 3, 6, 2, 8, 6 };  Character chs[] = { 'A', 'r', 'V', 'w' };  MyClass<Integer> iob = **new** MyClass<Integer>(inums);  MyClass<Character> cob = **new** MyClass<Character>(chs);  System.*out*.println("Max value in inums: " + iob.max()); System.*out*.println("Min value in inums: " + iob.min());  System.*out*.println("Max value in chs: " + cob.max()); System.*out*.println("Min value in chs: " + cob.min());  }  } |

***Bài 14.*** Bài tập về Collection dùng TreeMap:

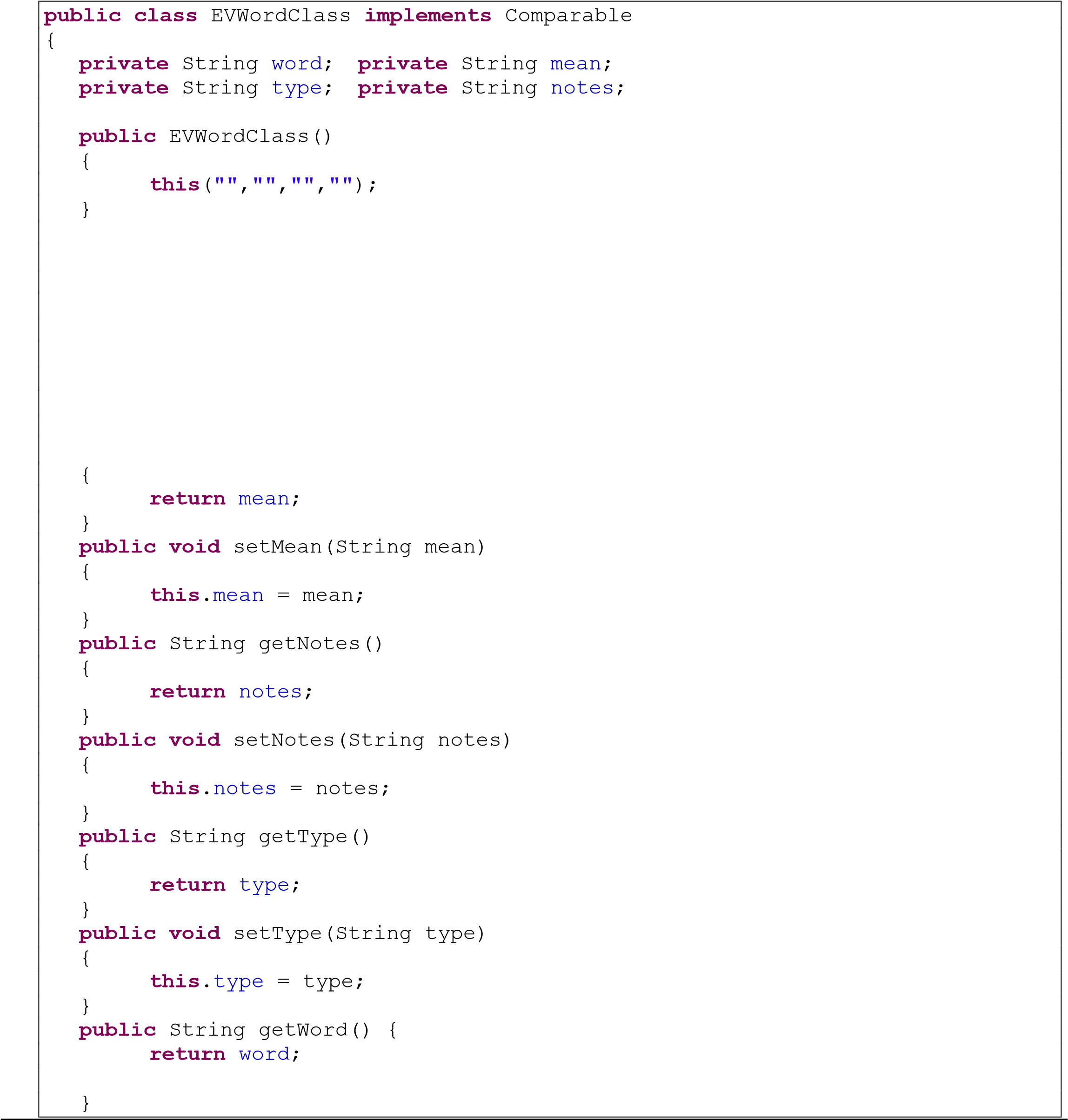
Viết 1 lớp mô tả 1 từ tiếng Anh bao gồm từ, nghĩa, loại từ, và phần ghi chú, lớp cần override phương thức *toString*, equals và phương thức so sánh 2 từ *compareTo* không phân biệt chữ thường hoa.

Lớp từ điển bao gồm các phuơng thức:

* + Thêm từ điển mới
  + Tra từ điển
  + *toString()* để in ra tất cả các từ trong từ điển

Lớp kiểm tra cho phép nhập vào các từ và tra cứu các từ đó *Hướng dẫn:*

Lớp từ bao gồm các thuộc tính(từ, nghĩa, loại từ, ghi chú), phương thức get/set, constructors, phương thức so sánh equals, toString

 **public** EVWordClass(String word, String mean, String type, String notes)

{

**this**.word = word; **this**.mean = mean; **this**.type = type;

**this**.notes = notes;

}

**public** String getMean()

|  |
| --- |
| **public** **void** setWord(String word) {  **this**.word = word;  }    **public** **boolean** equals(Object obj)  {  EVWordClass w = (EVWordClass)obj; **return** word.equalsIgnoreCase(w.getWord());  }    **public** String toString()  {  **return** word + "; " + type + "; " + mean + "; " + notes;  }    **public** **int** compareTo(Object o)  {  **return** **this**.word.compareToIgnoreCase(((EVWordClass)o).getWord());  }  } |

Lớp từ điển sử dụng Collections TreeMap

|  |  |  |  |
| --- | --- | --- | --- |
|  | { | **import** java.util.TreeMap; **public** **class** EVDictionary  **public** TreeMap<String,EVWordClass> dic;    // Tao TreeMap bao gom tu va **public** EVDictionary()  { dic = **new** TreeMap<String,EVWordClass>();  }    // Them tu moi vao tu dien **public** **boolean** addWord(EVWordClass word)  {  **if**(dic.put(word.getWord().toLowerCase(),word) != **null**)  **return** **false**; **return** **true**; | |
|  |  | }  // Tr **publi**  { }  **publi**  { | a tu  **c** EVWordClass lookup(String word) **return** dic.get(word);  **c** String toString()  String ret = "";  **for**(EVWordClass w:dic.values()) ret += w.toString()+"\n"; **return** ret; |
|  | }  } | | |

Lớp kiểm tra chương trình

**public** **class** EVDictionaryTest

{

**public** **static** **void** main(String[] args)

{

EVDictionary dic = **new** EVDictionary(); **for**(**int** i=1; i<10; i++)

{

dic.addWord(**new** EVWordClass("Word" + i, "", "Tu thu " + i, ""));

}

System.*out*.println(dic);

//Them tu

EVWordClass w = **new** EVWordClass("Word2", "", "Tu thu ", "");

**if**(!dic.addWord(w))

System.*out*.println("Khong them duoc!");

//Tra tu

EVWordClass l = dic.lookup("word2");

**if**(l != **null**)

System.*out*.println(l.toString());

}

}

***Bài 15.*** Một quân bài trong bộ bài gồm hai thuộc tính loại bài (cơ, rô, chuồn, bích) và thứ tự quân bài(2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A). Dùng Generic.

Viết các lớp sau:

* Lớp mô tả quân bài
* Lớp tạo bộ bài 52 quân bài không trùng nhau với phương thức là xào bài (dùng *shuffle()*)
* Lớp kiểm tra

*Hướng dẫn:*

Lớp mô tả quân bài

**public** **class** Card

{

**private** **int** face; **private** **int** suit;

**public** Card()

{

**this**.face = (**int**)(Math.*random*()\*4);

**this**.suit = (**int**)(Math.*random*()\*13);

}

**public** **int** getFace()

{ **return** face;

}

**public** **void** setFace(**int** face)

{

**this**.face = face;

}

**public** **int** getSuit()

{ **return** suit;

}

**public** **void** setSuit(**int** suit)

{

**this**.suit = suit;

}

**public** **boolean** equals(Object obj)

{

Card c = (Card)obj;

**return** (**this**.suit == c.getSuit() && **this**.face == c.getFace());

}

// Ham override

**public** String toString()

{

String ret = ""; **switch**(suit)

{

**case** 0: ret = "At";**break**; **case** 1: ret = "Hai";**break**; **case** 2: ret = "Ba";**break**; **case** 3: ret = "Bon";**break**; **case** 4: ret = "Nam";**break**; **case** 5: ret = "Sau";**break**; **case** 6: ret = "Bay";**break**; **case** 7: ret = "Tam";**break**; **case** 8: ret = "Chin";**break**; **case** 9: ret = "Muoi";**break**; **case** 10: ret = "J";**break**; **case** 11: ret = "Q";**break**; **case** 12: ret = "K";**break**;

}

**switch**(face)

{

**case** 0: ret += " Co";**break**; **case** 1: ret += " Ro";**break**; **case** 2: ret += " Chuon";**break**;

**case** 3: ret += " Bich";**break**;

}

**return** ret;

}

}

Lớp mô tả bộ bài

|  |
| --- |
| **import** java.util.ArrayList; **import** java.util.Collections;  **public** **class** Card\_Pack  {  **private** ArrayList<Card> pack; **public** Card\_Pack()  {  pack = **new** ArrayList<Card>();  **int** count =0;  **do**  {  Card c = **new** Card();  **if**(!pack.contains(c))  {  pack.add(c);  count ++;  }  }**while**(count<52);  }    **public** **void** shuffleCardPack()  {  Collections.*shuffle*(pack);  }  **public** String toString()  {  String ret = ""; **for**(Card c:pack) ret += c.toString() + "\n";  **return** ret;  }  } |

Lớp kiểm tra chương trình

**public**

**class**

CardTesting

{

**publi**

**c**

**static**

**void**

main(String[] args)

{

Card\_Pack cp=

**new**

Card\_Pack();

System.

*out*

.println(cp);

System.

*out*

.println(

"\nBAI SAU KHI XAO: \n"

;

)

cp.shuffleCardPack();

System.

*out*

.println(cp);

}

}

**Tuần 9. THAO TÁC NHẬP XUẤT DỮ LIỆU LƢU TRỮ TẬP TIN**

**Chƣơng 7. Nhập xuất trên Java** *Mục tiêu:*

* *Hiểu và áp dụng được đóng mở tập tin, thư mục*
* *Hiểu và áp dụng được các thao tác với tập tin, thư mục*

*Yêu cầu:*

* *Tất cả các bài tập lưu trong thư mục: D:\MaSV\_HoTen\Tuan09\*
* *Máy tính phải được cài đặt sẵn JDK (Java Development Kit)*
* *Máy tính phải có sẵn phần mềm soạn thảo hỗ trợ cho lập trình hướng đối tượng dùng ngôn ngữ lập trình Java (Eclipse/JCreator/NetBeans)*

***Bài 1.*** Viếtchương trình nhập vào một chuỗi và in ra chuỗi nghịch đảo của chuỗi nhập (Dùng BufferedReader và InputStreamReader).

***Bài 2.*** Hiển thị nội dung của một file tên test.txt lưu tại D:\test.txt Dùng BufferedInputStream thao tác đọc tập tin.

|  |
| --- |
| **import** java.io.\*;  **public** **class** BufferedFileApp  {  **public** **static** **void** main(String args[]) **throws** IOException  {  BufferedInputStream bStream = **new** BufferedInputStream( **new** FileInputStream("D:\\test.txt")); **int** ch=0;  **while** ((ch=bStream.read())!= -1)  {  System.*out*.print((**char**)ch);  }  bStream.close();  }  } |

***Bài 3.*** Copy nội dung một file text đến một file text khác. (Dùng

BufferedInputStream/BufferedOutputStream Hoặc dùng FileInputStream/FileOutputStream)

|  |
| --- |
| **import** java.io.\*;  **public** **class** CopyFileApp  {  **public** **static** **void** main(String args[]) **throws** IOException  {  **if** (args.length!=2)  {  System.*out*.println("Usage : java CopyFileApp <SrcFile> <DestFile>"); **return**;  }    String SourceFile=args[0]; // tập tin nguồn  String DestFile =args[1]; // tập tin copy  // Tạo bộ đệm đọc dữ liệu từ tập tin nguồn  BufferedInputStream inFile = **new** BufferedInputStream(  **new** FileInputStream(SourceFile));  // Lấy kích thước tập tin nguồn **int** FileSize = inFile.available();  // Tạo bộ đệm ghi dữ liệu vào tập tin đích  BufferedOutputStream outFile = **new** BufferedOutputStream( **new** FileOutputStream(DestFile));  // Chuyển dữ liệu **int** ch=0; **while** ((ch=inFile.read())!= -1)  {  outFile.write(ch);  }    System.*out*.println(FileSize + " bytes da duoc copy xong.");    inFile.close(); outFile.close();  }  } |

***Bài 4.*** Dùng DataOutputStream và DataInputStram để ghi và đọc những kiểu dữ liệu khác nhau trên file.

***Bài 5.*** Liệt kê danh sách các thư mục con và tập tin của 1 thư mục. Nếu thư mục, hiển thị thêm <DIR> phía trước của tên.

|  |
| --- |
| **import** java.io.\*; **public** **class** FileApp  {  **public** **static** **void** main(String args[]) { |
| File curDir=**new** File("C:\\");  String[] dirs=curDir.list();    **for** (**int** i=0; i<dirs.length; i++)  {  File f=**new** File("C:\\"+dirs[i]);  **if** (f.isDirectory())  {  System.*out*.println("<DIR> "+dirs[i]);  } **else**  {  System.*out*.println(" "+dirs[i]);  }  }  }  } |

***Bài 6.*** Truy cập ngẫu nhiên trên file, viết chương trình ghi 6 số kiểu double xuống file, rồi đọc lên theo thứ tự ngẫu nhiên

***Bài 7.*** Thực hiện đọc ghi đối tượng dùng ObjectInputStream và ObjectOutputStream.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **import import public**  { **public**  { | java.io.\*; java.util.\*;  **class** ObjectWriteExApp  **static** **void** main(String args[])  **throws** IOException  ObjectOutputStream oStream = **new** ObjectOutputStream( **new** FileOutputStream("Container.txt"));  // ghi đối tượng String  oStream.writeObject(**new** String("Hello World"));  // ghi đối tượng Fruit oStream.writeObject(**new** Fruit("Orange",10)); oStream.writeObject(**new** Fruit("Apple",5));  // ghi đối tượng Date oStream.writeObject(**new** Date());  System.*out*.println("Ghi 4 doi tuong vao tap tin Container.txt"); | |
|  | | }  }  **class** Fruit **implements** Serializable  {  String name=""; **int** weight=0;    **public** Fruit(String n,**int** w){ name =n; | | |
| weight=w;  }  **private** **void** writeObject(ObjectOutputStream out) **throws** IOException{ out.writeObject("X "+name); out.writeInt(weight-1);  }  **private** **void** readObject(ObjectInputStream in) **throws** IOException,ClassNotFoundException{ name =(String)in.readObject(); weight=in.readInt();  }  **public** String toString()  {  **return** (name +" "+ weight+" g");  }  } | | |

***Bài 8.*** Dùng BufferedReader đọc từng ký tự từ Console. Việc đọc kết thúc khi gặp dấu chấm (dấu

chấm để kết thúc chương trình).

***Bài 9.*** Dùng BufferedReader đọc chuỗi ký tự từ Console. Chương trình kết thúc khi gặp chuỗi đọc là chuỗi “stop”.

***Bài 10.*** Định nghĩa lớp lưu trữ thông tin của sinh viên. Cho phép nhập dữ liệu sinh viên và lưu trữ thành file data.dat. Đọc dữ liệu từ tập tin, đưa vào mảng và hiển thị kết quả. Thông tin của SV bao gồm mã, họ tên, địa chỉ, số điện thoại và điểm trung bình của năm học vừa qua.