#### Review :

#### OOP Concepts:

#### Encapsulation

#### Inheritance

#### Polymorphism

#### Encapsulation

#### Class = Data (fields/properties) + Methods

#### Data of a class should be hidden from the outside. Access modifier: public, protected, default, private

#### All behaviors should be accessed only via methods.

#### A method should have a *boundary condition:* Parameters must be checked (use if statement) in order to assure that data of an object are always valid.

#### Constructor: A special method it’s code will execute when an object of this class is initialized.

#### Getters/setters

#### Access modifier : public, protected, default, private

**Int ID\_Num**

**String Name**

**YearOfBirth**

**Address**

**PERSON()**

**PERSON(, , , )**

***getID\_Num()***

***setID\_Num(newID)***

**input(){ Scanner sc=new Scanner();… }**

**PERSON**

#### OOP Concepts:

#### Inheritance and Polymorphism

**By the end of this week, you will be able to...**

* Explain the value of **inheritance** as a feature in object oriented programming languages
* Use the keyword **extends**
* Explain the relationship between a **superclass** and a **subclass**
* Use **UML Diagrams** to display class hierarchies
* Explain an **“is-a”** /**”has-a**”relationship between classes
* Describe how **Java object construction** occurs from the inside out
* Use same-class and **superclass** **constructors** in class creation
* Create methods which **override** from a superclass
* Contrast method **overloading** and method **overriding**
* Explain the purpose of **polymorphism**
* Step through decisions made at **compile time** and **runtime**
* Use **casting** of objects to aid the compiler
* Use the keyword "**abstract**"
* Compare and contrast “**inheritance of implementation**” and “**inheritance of interface**”
* Decide between **Abstract Classes** and **Interfaces**

**Let’s begin**

**Question: Why is used** Inheritance?

Ability allows a class having members of an existed class 🡺 Re-used code, save time

Example 1: Write a program to manager some students in FPTU. At FPTU, there are some software engineer students (SE\_Student) and business students(SB\_Student).

SE\_Student includes : RollNum, Name, LabMark, PracExamMark

SB\_Student includes : RollNum, Name, MathMark, Skill (Skill is one in excel, word, powerpoint or empty string)

When the program runs, it creates two SE\_Student objects and one SB\_student object. And then it display information those objects.

The output will be showed like:

SE Students:

<<RollNum – Name –LabMark-PEMark- AvgMark – Status>>

<<RollNum – Name –LabMark-PEMark- AvgMark – Status>>

SB Students:

<<RollNum – Name –MathMark- Skill – Status>>

*Hint:*

*1-Lan-7-8-7.5-passed*

***AvgMark = (LabMark+PEMark)/2 or avgMark=MathMark + bonus (if skill is not empty, bonus =10%MathMark)***

father/superclass

* RollNum
* Name

STUDENT()

STUDENT(RollNum, Name)

*Getters  
Setters*

*InputStudent()*

*OutputStudent()*

**STUDENT**

***“is a” relationship***

***“is a” relationship***

* MathMark
* Skill

SB\_STUDENT()

SB\_STUDENT(,,,)

*getters*

setters  
inputSBStudent()

outputSBStudent()

**SB\_STUDENT**

* LabMark
* PEMark

SE\_STUDENT ()

SE\_STUDENT(,,,)

*getters*

setters

inputSEStudent()

outputSEStudent()

**SE\_STUDENT**

**Son/subclass**

**Mean: A SE\_Student object is a Student object  
 A SB\_Student object is a Student object  
 Student is generation of SE student or SB student**

**Implements: next lesson**

**Example 2: Write a program to manage some circle objects. A circle object includes: x, y, radius. When the program runs, it creates 2 circle objects and so a message “two outer contact circles” if distance of their point equals sum of their radius.**

**Class Circle: x,y,radius=> te**

* x
* y

Point()

Point(x,y)

*getters  
setters*

**Point**

**Has a**

* Point p
* float radius

Circle()

Circle(,,,)

*getters  
setters*

**Circle**

**Implements: next lesson**

**Extension:**

**Class Student { int id,String name, Course [] array…}  
Class Course { id, name, credit, Student[] array}**

**Example3: How to detect father/son classes? Draw UML design?**

Electric Products< code, name, make, price, warranty, voltage, power>

Ceramic Products < code, name, make, price, type >

Food Products < code, name, make, price , date, expiredDate >

**Ve UML Desin (ve quan he giua cac class)**

**In this assignment, leaners have to draw UML design and implement these classes. When the program runs show a menu:**

1. **create an electric product**
2. **create an ceramic product**
3. **create an food product**
4. **display three products**

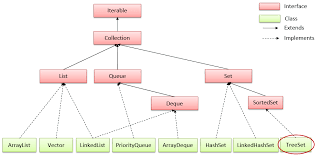
**Extension:**

**Consider statement Student x= new Student(1,”John”);  
 SEStudent y=new SEStudent(2,”Mia”,7,5);  
 SBStudent z=new SBStudent(3,”Maica”,7,”excel”);**

**So,**

**Student temp=y;**

**-------------------------------------------------------------------------------------------------------------------**

****

**TreeSet: Node (root, fatherNode, childNode, LeafNode)**

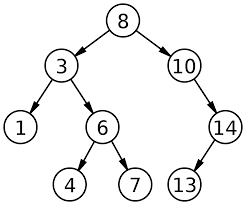
**Node: 3 parts : Left, Data, Right ; Data: int, String, Object : Book : id, name, price**

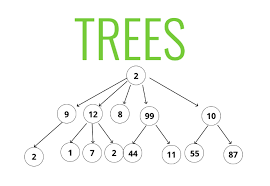
**Depth**

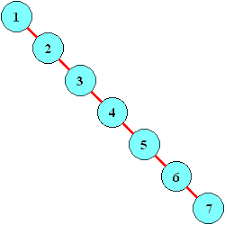
**LNR: 1,3,4,6,7,8,10,13,14**

**RNL: 14, 13,10,8,7,6,4,3,1**

**Breath: => level order Queue:8,3,10,1,6,14,4,7,13,**

** Binary Tree, BSearchT. Balance Tree. Add : 12, abc, 100**

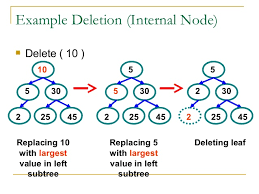
**B-Tree**

****

**Insert: 8, 3, 1, 6,4,7 ,10,14,13**

**1, 3, 6,4,7,…… Search : 13 => balance**

**Delete:**

****

**Traversal: Depth, Breath**

**Example : Depth**

**LNR: 2, 5,10,25,30,45**

**---------------------------**

**HashSet<T> h=new HashSet();**

|  |  |
| --- | --- |
| **Hash Index** | **Diachi** |
| **0** | **Null** |
| **…** | **…**  240: ant  200:apple |
| **19** | **200**  230:baby |
| **20** | **230**  250:bad |
| **21** | **270**  270:cook |
| **…** | **…** |
| **25** | **null** |

**Object: hashCode(); equals(); wait();….**

**Add(Element): java use cong thuc (hash function) de tinh index phu hop**

**Vi du: Them 1 element**

**add(“apple”) : 97%26= 19 (hash function -> index)  
 add(“baby”): 98%26=20   
 add(“bad”): 98%26=20**

**add(“ant”): 97%26= 19 (hash function -> index)**

**add(“cook”): 99%26=21-> index**

**add(“cook”);**

**add(“baby”) => java tu choi them “baby” vi bi trung(duplicate) : bang cach goi ham equals() de check**

**Find an element**

**contains(“ant”);97%26=19=> chi find tren nhom element tai vi tri 19**

**Sort elements   
 1. Copy elements to array  
 2. Use Arrays.sort de sap xep array truyen thong ( fixed size)**

**Interface MAP <K-Object,V-Object>: put(), clear(), get(key), keySet(), values(), iterator()  
classes: HashMap or TreeMap**

**Vi du 1:**

**Tuong ung voi moi “Ten campus” , thi xac dinh dc ds cac sv cua campus do**

**HashMap <String,ArrayList<Student>>**

|  |  |
| --- | --- |
| **Key** | **Value** |
| **“SG”** | **{ {1,”a”},{“2”,”v”},{3,”d”}}** |
| **“HN”** | **{ {3,”sd”},{“4”,”as”},{5,”fg”}}** |
|  |  |

**Vi du2: Su dung data structure de luu “ten san bay” : A va ds cac san bay co mo chuyen tu san bay A  
HashMap<String,ArrayList<String>>**

|  |  |
| --- | --- |
| **Key** | **Value** |
| **“Tan Son Nhat”** | **{ “Noi Bai”, “Da Nang”,”Cat Bi”}** |
| **“Noi Bai”** | **{“Lien Khuong”,”Tan Son Nhat”, “Con Dao”}** |

**Vi du3:   
*can data structure de luu thong tin dia chi IP va ten cac resource ma dia chi IP do connect***

***127.0.0.1 http://www.vnexpress.vn 200 ok 23/2/2020  
 127.2.3.1*** [***http://www.tuoitre.vn***](http://www.tuoitre.vn) ***404 fail  
 127.0.0.1 htttp://www.nhac.vn 200 ok  
…..***

**HashMap<String, ArrayList<String>>  
key: IP, value : {ten cac trang web}**

|  |  |
| --- | --- |
| **Key** | **Value** |
| **127.0.0.1** | **{** “**http://www.vnexpress.vn” , “htttp://www.nhac.vn “}** |
| **127.2.3.1** | **{“**[**http://www.tuoitre.vn**](http://www.tuoitre.vn)**”}** |

**Caution: key: ko trùng,   
 value: có thể trùng  
HashMap<String,ArrayList<String>>**

|  |  |
| --- | --- |
| **Key** | **Value** |
| **“a”** | **~~{“an”,”ant”,”apple”}~~ {“apple”,”area” }** |
| **“b”** | **{“baby”,”bad”,”book”,”beer”}** |
| **“c”** | **{“cook”,”can”,”could”}** |

1. **Them 1 entry : <String, ArrayList<String>>**

**Put (“a”,{“an”,”ant”,”apple”});**

**Put (“b”,{“baby”,”bad”,”book”,”beer”})**

**Put (“c”,{“cook”,”can”,”could”});**

**Put (“a”,{“apple”,”area” }); //neu them entry nay, no se thay the value cua key bi trung**

1. **Lấy Value khi biết key**

**List<String> list= get(“b”); => list ={”baby”,”bad”,”book”,”beer”}**

1. **Lấy ds các key**

**Set<String> keys=keySet(); => keys : {“a”,”b”,”c”}**

1. **Lấy ds cac value**

**List<ArrayList<String>> result=values(); =>result: {  
 {“an”,”ant”,”apple”}, {“baby”,”bad”,”book”,”beer”},   
 {“cook”,”can”,”could”} }**

1. **Update   
   5.1 : Update value của 1 key**vi du: danh sach newList: {“cooker”,”could” } thay the cho value cu cua key “c”  
   => ham put(“c”, newList)

5.2: Update “baby” cua key “b” thanh “bom”  
- ArrayList<String> list=get(“b”) => list: {“baby”,”bad”,”book”,”beer”}  
- tim vitri cua “baby” trong list pos=list.find(“baby”)=> pos:0  
- list.set(pos,”bom”) , list :{ “bom”,”bad”,”book”,”beer”}  
- put(“b”,list)

**5.3: Đưa thêm “ban” vào value của key “b”**  
- ArrayList<String> list=get(“b”) => list: {“baby”,”bad”,”book”,”beer”}  
- list.add(“ban”) => list: {“baby”,”bad”,”book”,”beer”,”ban”}  
- put(“b”,list)

**6.Xóa**

**6.1 Xóa 1 entry khi biết key  
 remove(“b”) => mất “b”,{“”,”,,,” }**

**6.2 Xóa 1 phần trong value khi biet key  
 xoa “cook” cua key “c”  
- ArrayList<String> list=get(“c”)=> list={“cook”,”can”,”could”}  
- tim “cook” trong list, pos=list.contains(“cook”)=> pos=0  
- list.remove(pos);**