**PRO192  
Lab3\_1: Inheritance**

**Learning Outcomes:**

Upon successful completion of this workshop, you will have demonstrated the abilities to:

* Design and implement classes in the “is-a” relationship.
* Practice casting
* Describe to your instructor what you have learned in completing this workshop.

**Requirements:**

**Part 1:** [5 points]

To complete this task you should read and study the lecture [Inheritance](file:///D:\FPT_University\Presentation\PRO192\PRO192-materials\index.html)

Step 1: Create a new project named “**ItemManager**”.

Step 2: Create a package named “**DTO**”, it contains some files: Item.java, Vase.java, Statue.java, and Painting.java

Step 3: Create another package named “**GUI**”, it contains the AntiqueShop.java file

Implement the class diagram as follows:

|  |
| --- |
| Item |
| Int |
| +Item() +Item(int, String) +getters/setters +output():void +input():void |

|  |
| --- |
| Vase |
| -height: int -material: String |
| +Vase() +Vase(int, String, int, String) +getters/setters +outputVase():void +InputVase(): void |

|  |
| --- |
| Statue |
| -weight: int -colour: String |
| +Statue() +Statue(int, String, int, String) +getters/setters +outputStatue():void +inputStatue():void |

|  |
| --- |
| Painting |
| -height: int -width: int -isWatercolour: boolean -isFramed: boolean |
| +Painting() +Painting(int, String, int, int, boolean, boolean ) +getters/setters +outputPaiting():void +inputPainting():void |

|  |
| --- |
| AntiqueShop |
|  |
| +main():void |

The AntiqueShop class is making use of Vase, Statue, and Painting, in the sense that it has declared references to them, and thus there is a dependency.

**Requirement**:  
 1. In the file Item.java,

* The method input(): Using Scanner class to input all fields of the Item class. Verify: value>0, creator is not empty
* The method output(): print out all fields

2. In the file Vase.java,

* The method inputVase(): Using Scanner class to input all fields of the Vase class.
* The method outputVase(): print out all fields of the Vase class

*Hint:*

public class Vase{  
 …

//this method is used to input all fileds of a vase object  
 public void inputVase(){  
 input(); // call the inherited method to input two fields: value, creator

//TODO: you is required to add more your code to input two fields : height, material

// use try..catch/throws to handle exceptions  
 }

//this method displays information of a vase object  
 public void outputVase(){  
 output(); // call the inherited method to print two fields out: value, creator

System.out.println(“Height:” + height);  
 System.out.println(“Material:”+ material);  
 }

…  
}

1. You do the same for Statue class, Painting class
2. In the file “AntiqueShop.java”. you type like as follow:

public class AntiqueShop {

public static void main(String[] args){  
 Item item=null;

int choice=0;

Scanner sc=….

do{  
   
 System.out.println(“1. Create a Vase:”);

System.out.println(“2. Create a Statue:”);

System.out.println(“3. Create a Painting:”);

System.out.println(“4. Display the Item:”);

System.out.println(“Input a choice:”);

Choice=sc.nextInt();  
 switch(choice) {  
 case 1:  
 item=new Vase();  
 ((Vase)item).inputVase();

break;   
case 2:  
 item =new Statue();  
 ((Statue) item).inputStatue();

break;

case 3:

item =new Painting();  
 ((Painting) item).inputPainting();

break;

case 4:  
 if(item!=null) {  
 if(item instanceof Vase)  
 ((Vase) item).outputVase();  
 else if(item instanceof Statue)  
 ((Statue) item).outputStatue ();  
 else if(item instanceof Painting)  
 ((Painting) item).outputPainting ();   
 }  
 else

System.out.println(“ you need to create an object”);

break;  
 }  
  
 }while(choice<=4); }

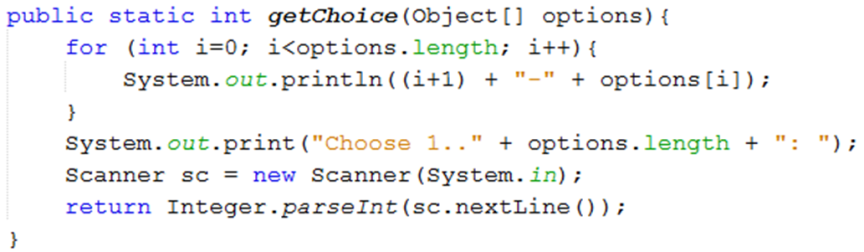
}

1. (Optional) Now, you is required to update the above program. You should create a new class named **Menu**. This class contains one static method

*//use this method to show pre-defined options*

*//input: an array contains the list of options*

*//output: return a user’s choice that is inputted from the keyboard.*



* Update the main method to use the Menu class.

public class AntiqueShop {

public static void main(String[] args){

String[] options={“ Create a Vase “,”Create a Statue”,” Create a Statue”,” display the item”};  
 Item item=null;

int choice=0;

do{  
   
 choice=Menu.getChoice(options);

switch(choice){  
 case 1:  
 item=new Vase();  
 ((Vase)item).inputVase();

break;   
 ….  
 }while(…);

}

**Lab3\_2: Polymorphism**

**Learning Outcomes:**

Upon successful completion of this workshop, you will have demonstrated the abilities to:

* Practice polymorphism.
* Understand the principles and the use of abstract classes and interfaces in Java
* Describe to your instructor what you have learned in completing this workshop.

**Part 1:** [5 points]

Using Java language to build an application. This app is an example to expose parts of your classes. Consider the class diagram as follows:

|  |
| --- |
| **<<abstract class>> Organization** |
| # size: int |
| +Organization () + Organization (String) + getSize():int + setSize(int):void + communicateByTool(): void, abstract + **toString**():String |

|  |
| --- |
| **University** |
| #name: String |
| +University () + University (int, String) + communicateByToTool(): void + enroll():void + educate(): void + **toString**():String |

|  |
| --- |
| **Colony** |
| # place: String (land/ocean) |
| +Colony () + Colony (int, String) + communicateByTool(): void + grow():void + reproduce():void + **toString**():String |

|  |
| --- |
| **<<interface>> Role** |
| + createWorker():void |

|  |
| --- |
| **FPTUniversity** |
| address: String |
| +FPTUniversity () + FPTUniversity (int, String, String) + gettter/setter + **toString**():String |

|  |
| --- |
| **BeeColony** |
| type: String (honey/wasp) |
| +BeeColony () + BeeColony (int, String, String) + gettter/setter + **toString**():String |

The *Organization* class contains the “communicateByTool()” method. This method describes the communication way between members. For now, we don’t have any information to implement it, so it should be the *abstract* method.

The *Colony* class and *University* class will extend the *Organization* class, they must override the “communicateByTool()” method.

The *BeeColony* class and *FPTUniversity* class have got quite different inheritance hierarchies. But both *create workers* and they don't share much in common. So, we declare an interface named “Role” that contains the “createWorker()” common method. These classes will implement it.

To complete this task you will implement the class structure above

Step 1: Create a new project named “OrganizationManager”.

Step 2: Create a package named “**DTO**”, it contains some files: Organization.java, Colony.java, University.java, BeeColony, and FPTUniversity.java

Step 3: Create another package named “**GUI**”, it contains the Tester.java file

**Requirements**:

1. In the file Organization.java

* Declare this class is abstract
* The method communicateByTool() is an abstract method.
* The method toString(): return the string as format:  
   “*the organization’s size is” + the value of the size field*

2. In the file Colony.java,

* This class extends the Organization class
* The method communicateByTool(): must be overridden to print out the string as *“the colony communicate by sound”*
* The method grow(): print out the string as “*an annual cycle of growth that begins in spring”*
* The method reproduce():(): print out the string as *“Colony can reproduce itself through a process”*
* The method toString(): return the string as format:  
   “*the colony size is” + the value of the size field + “, the colony’s place is” + the value of place field*

3. In the file BeeColony.java,

* This class extends the Colony class and implements the Role interface
* The method toString(): return the string as format:  
   “*the colony’s type is ” + the value of the type field+ “, size is about” + the value of the size field + “, and the place is” + the value of place field*
* The createWorker() method: must be overridden to print out the string as “Worker bees perform all the work of the bees”

4. In the file University.java,

* This class extends the Organization class
* The method communicateByTool(): is overridden to print out the string as *“in the university, people communicate by voice”*
* The method enroll(): print out the string as “*The registration for enrollment is only valid when the University has received all enrollment documents and enrollment fees”*
* The method educate(): print out the string as *“provide education at university standard”*
* The method toString(): return the string as format: “encourage the advancement and development of knowledge”

5. In the file FPTUniversity.java,

* This class extends the University class and implements the Role interface
* The method toString(): return the string as format: “FPTU has four campuses Hanoi, HCM, DaNang, CanTho, QuyNhon”
* The createWorker() method: must be overridden to print out the string as “providing human resources”

6. In the file Tester.java, you type:

public class Tester{

public static void main(String[] args){  
  
 **Colony obj1=new BeeColony(2000, “honey”, “land”);** System.out.println(obj1);  
 obj1.grow();  
 obj1. reproduce();

**University obj2=new FPTUniversity(100000, “FPT”, “Cantho”);**  System.out.println(obj2);obj2.enroll(); obj2.educate();  
   
 **Role** **df**= **new BeeColony(3000, “wasp”, “land”);** System.out.println(df);  
 df.createWorker();

**df**= **new FPTUniversity(100000, “FPT”, “Hanoi”);** System.out.println(df);  
 df. createWorker ();  
   
 }

}