

Workshop #4: 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: [7 points]

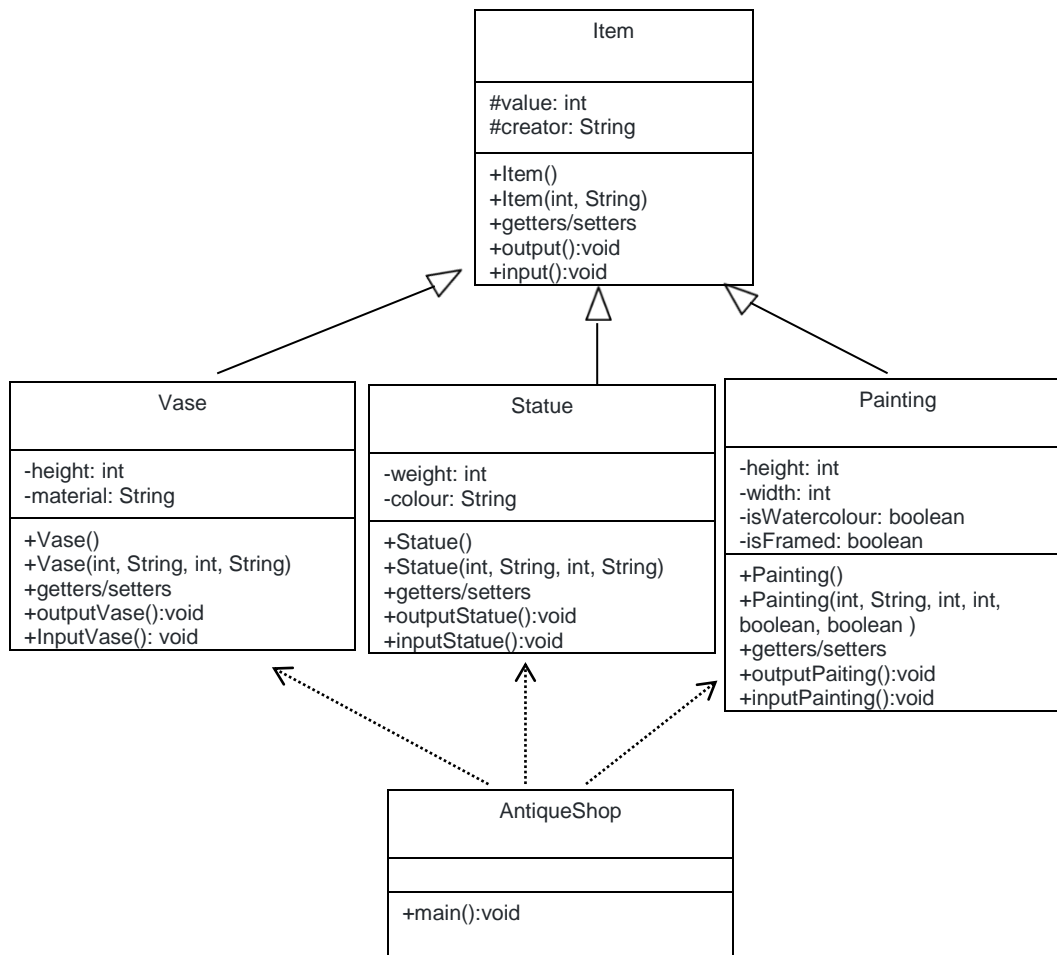
To complete this task you should read and study the lecture [Inheritance](#)

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:



This is an “**association**” relationship and simply denotes that **AntiqueShop** 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{
    ...
    public void inputVase(){
        input(); // call the inherited method to input two fields: value, creator
        Scanner sc=new Scanner(System.in);
        System.out.println("Input a height:");
        height=sc.nextInt();
        System.out.println("Input a material:");
        sc=new Scanner(System.in);
        material =sc.nextLine();

    }
    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);
    }
    ...
}
```

3. You do the same for Statue class, Painting class

4. In the file "Test.java". you type like as follow:

```
public class Tester{

    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);

    }

}
```

Part 2: Draw the memory map when the program runs [3 points]

Explain step by step what happened when the program runs and answer some questions.

- What is stored in the static heap, stack, dynamic heap?
- What are objects in the program?
- What is the item variable storing?
- Why must you cast to call the method inputVase()/outputVase()?
- What is the error thrown when you cast it wrong?
- What methods can you call if you don't cast the item variable?