Workshop #5: **Polymorphism**

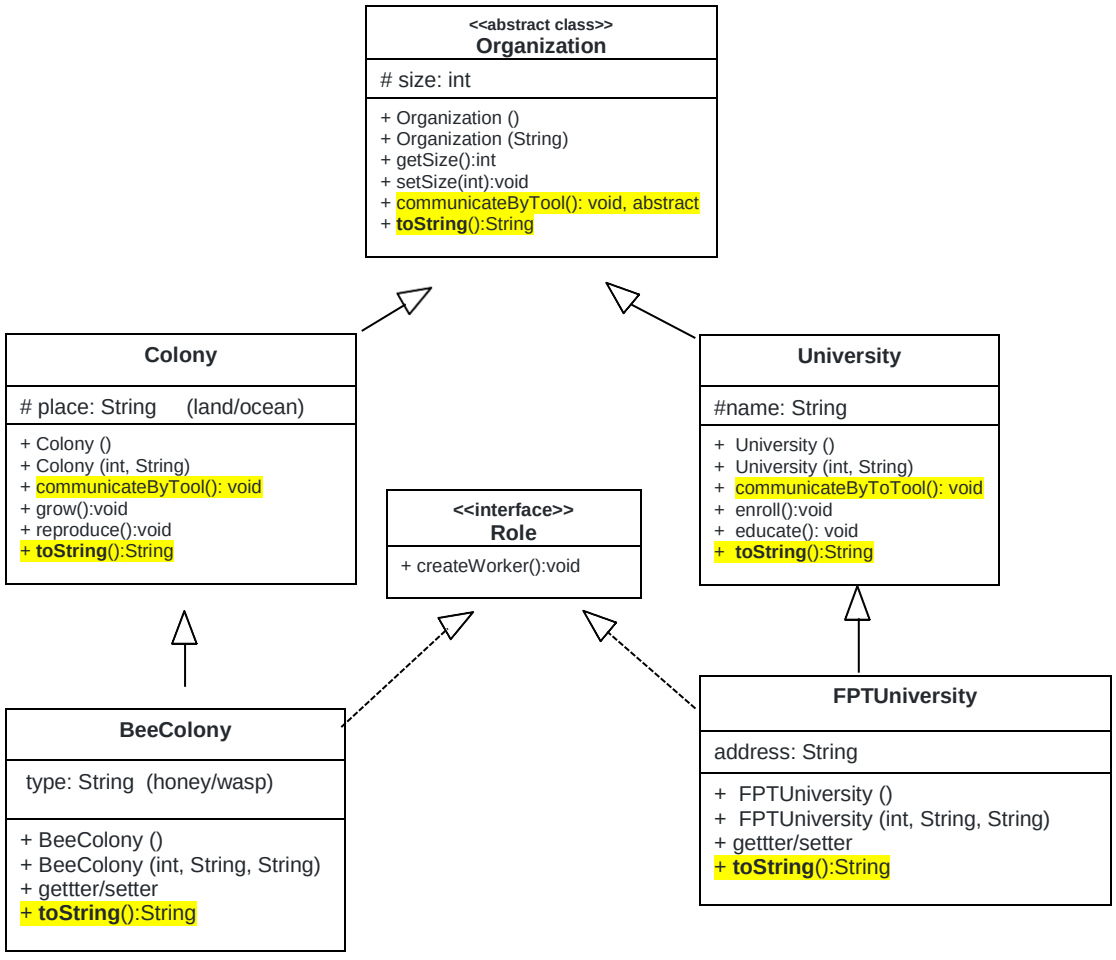
Learning Outcomes:

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: [7 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:



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 ();

  }
```

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?
- Why the Organization class is abstract?
- Why must the Colony/University class implement the communicateBytool() method?
- You explain the polymorphism feature in your code.
- Describe the difference between an interface and an abstract class.