**1.**

* **static heap**: is the memory for your variables allocated when the program starts.
  + Java Runtime Classes
  + Code of Program
  + Tester Class
  + Organization, Colony, University, BeeColony, FPTUniversity Classes

* **stack**: used to store local
  + main
  + args[]
  + obj1, Names of Attributes and Methods of obj1
  + obj2, Names of Attributes and Methods of obj2
  + df, Names of Attributes and Methods of df
  + du, Names of Attributes and Methods of du
* **dynamic heap**: used to allocate memory for local variables, function parameters.
  + obj1 (2000,"honey","land")
  + obj2 (10000,"FPT","Cần Thơ")
  + df (3000,"wasp","land")
  + du (10000,"FPT","Hà Nội")

**2.**

Because we declare it as an object.



Organizations are abstract because they have no entities, they have many abstract methods, or even concrete methods.

**3.**

Both "Colony" and "University" have the same properties as "communicateByTool", so we need to write Organization as an Abstract and let the Colony/University class implement it.

**4.**

Polymorphism means "many forms", and it happens when we have many classes that are related by inheritance.

Here, the superclass called "Organization" has a method called "communicateByTool".

Subclasses of "Organization" such as "Colony" and "University" have separate ways of communicatingByTool.

**5.**

|  |  |
| --- | --- |
| Abstract class | Interface |
| Abstract class can have abstract and non-abstract methods. | Interface can have only abstract methods. Since Java 8, it can have default and static methods also. |
| Abstract class doesn't support multiple inheritance. | Interface supports multiple inheritance. |
| Abstract class can have final, non-final, static and non-static variables. | Interface has only static and final variables. |
| Abstract class can provide the implementation of interface. | Interface can't provide the implementation of abstract class. |
| The abstract keyword is used to declare abstract class. | The interface keyword is used to declare interface. |
| An abstract class can extend another Java class and implement multiple Java interfaces. | An interface can extend another Java interface only. |
| An abstract class can be extended using keyword "extends". | An interface can be implemented using keyword "implements". |
| A Java abstract class can have class members like private, protected, etc. | Members of a Java interface are public by default. |