Object Oriented Programming

ICT2122

Abstraction

P.H.P. Nuwan Laksiri
Department of ICT
Faculty of Technology
University of Ruhuna

Lesson 03 - OOP Concepts - Part 03

Recap

- Polymorphism
- Method Overloading
- Method Overriding
- Dynamic Polymorphism
- Static Polymorphism

Outline

- Abstraction
- Abstraction in JAVA
- Abstract Methods
- Abstract Classes
- Hands-On

Object Oriented Concepts

- Object Oriented Programming simplifies the software development and maintenance by providing some concepts,
  - Object
  - Class
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation

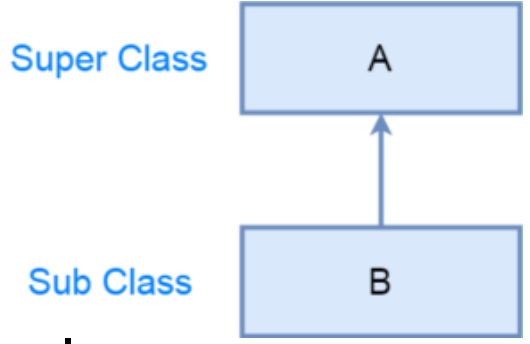
Classes and Objects

A class is like a cookie cutter; it defines the shape of objects

Objects are like cookies; they are instances of the class

Photograph courtesy of Guillaume Brialon on Flickr.

Inheritance



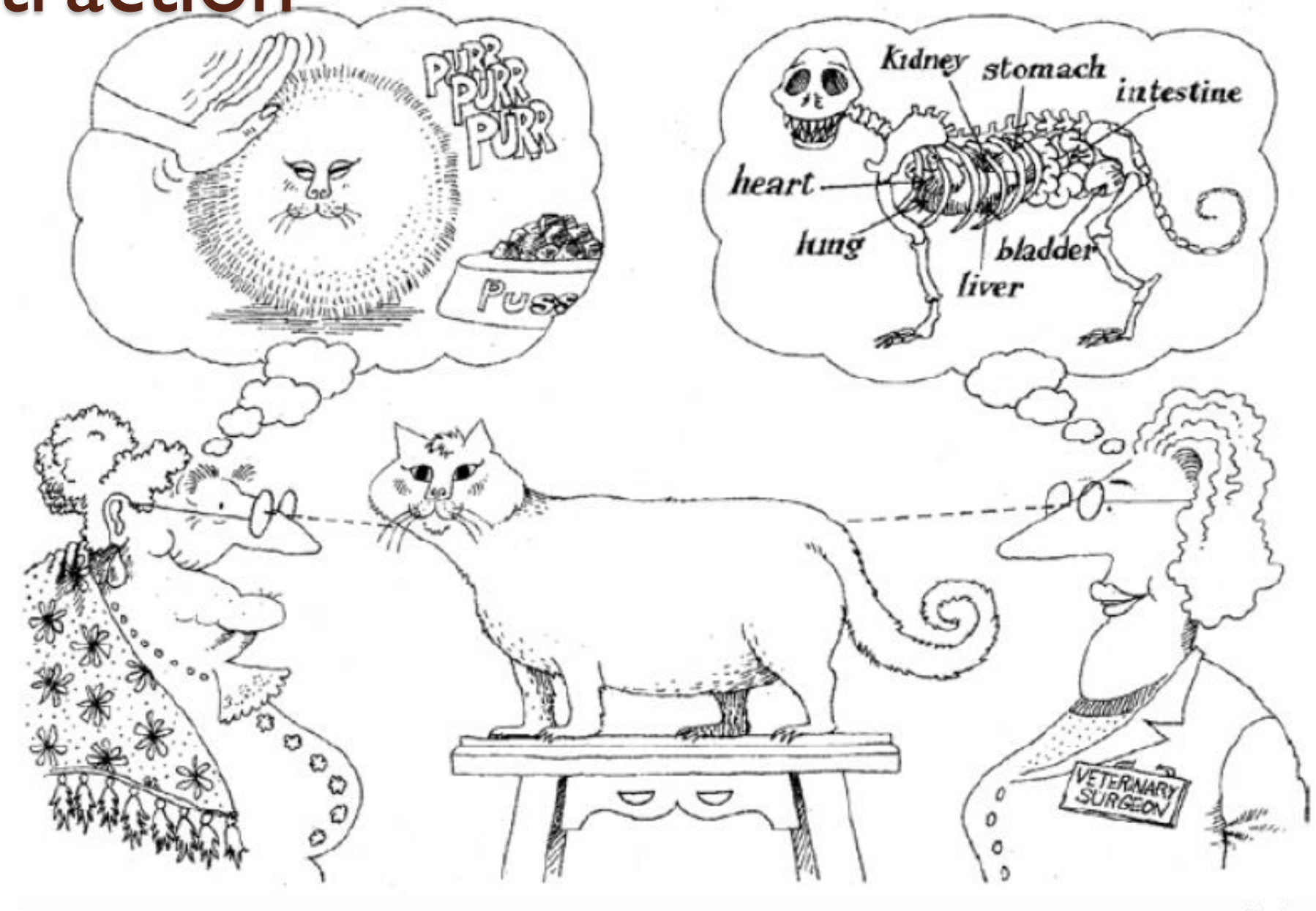
- Inheritance is a mechanism that allows
  - a subclass to inherit the properties and behaviors of a superclass.
- This means that the
  - subclass can access and use all the methods and variables of the superclass,
  - as well as add its own methods and variables.
- The subclass can also
  - override methods from the superclass to provide its own implementation.
- Inheritance enables
  - code reuse and makes it easier to manage and maintain complex systems
  - by reducing duplication and
  - providing a hierarchical structure for classes.
- It is a key feature of object-oriented programming and is widely used in Java

Polymorphism

- Poly-Morphism-> ability to have multiple forms (shapes) of the same thing.

- Polymorphism is the capability of an action or method to do different things based on the object that it is acting upon.

Abstraction



Abstraction

- "We (humans) have developed an exceptionally powerful technique for dealing with complexity.

  We abstract from it.

- Unable to master the entirety of a complex object, we choose to ignore its inessential details, dealing instead with the generalized, idealized model of the object"

- (Ref:Wulf)

Abstraction

- Extract only the necessary details.
- Remember the Cat example above ; how a nonmedical person model a Cat and how a medical person model a Cat.
- Class represents a real-world entity.
  - class **contains only the essential details matching to the problem domain**.

Abstraction in JAVA

- Abstraction in Java is a mechanism that helps to reduce the complexity of a system by **hiding its implementation details from the user**.
- This means that the user only sees what is necessary to perform a certain task and does not need to know about the underlying implementation.

- For example,
  - sending a SMS, you just type the text and send the message.
  - You don't know the internal processing about the message delivery.

Abstraction in JAVA

- Abstraction can be achieved in Java using abstract classes and interfaces.
  - Abstract class (0 to 100%)
  - Interface (100%).

Abstract Method

- A method that is **declared as abstract** and **does not have implementation** is known as abstract method.

abstract void printStatus();

Note followings

- no body and
- abstract

Abstract Class

- An abstract class is a class that **cannot be instantiated**
  - but **can be extended** by other classes.
- An abstract class can have **both abstract and concrete methods**.
- Abstract methods are methods that have no implementation and must be overridden by any concrete (non-abstract) subclass..
- All other functionality of the class still exists, and its fields, methods, and constructors are all accessed in the same manner.

Abstract Class

- Use the "**abstract**" keyword to declare a class abstract.

- The keyword abstract appears in the class declaration somewhere before the "class" keyword.

```
public abstract class Employee
```

Abstract Class – Hands-On

```
public abstract class Bike {
        abstract void run();
}

public class Honda extends Bike {
        void run()
        {
                System.out.println("running safely..");
        }
        public static void main(String args[])
        {
                Bike obj = new Honda4();
                obj.run();
        }
}
```

Abstract Class - Hands-On

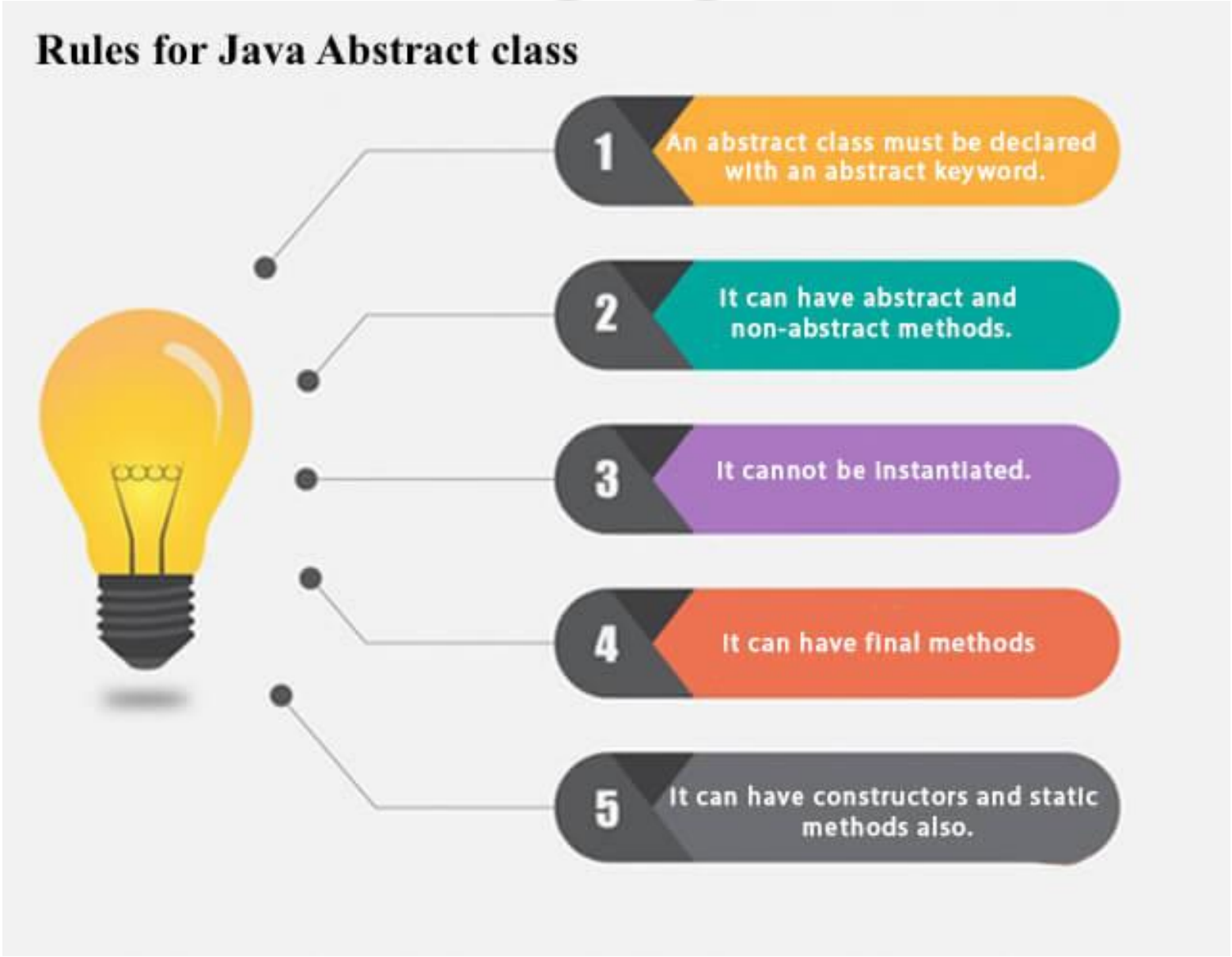
```
public abstract class Bike{
        Bike() //Constructor
        {
                System.out.println("bike is created");
        }
        abstract void run();  //abstract method
        void changeGear() //concrete method
        {
                System.out.println("gear changed");
        }
}

public class Honda extends Bike{
        void run()
        {
                System.out.println("running safely..");
        }
}
```

Abstract Class - Hands-On

```
 class TestAbstraction
{
        public static void main(String args[])
       {
                Bike obj = new Honda();
                obj.run();
                obj.changeGear();
       }
}
```

Abstract Class - Highlights



Abstraction

- If there is **any abstract method** in a class, that **class must be abstract**.

- If you are **extending any abstract class** that have abstract method, **you must either provide the implementation of the method or make this class abstract.**

Abstraction - Homework

- Identify the difference between “Abstract Classes” and “Concrete Classes”.
- Can a abstract class have a constructor? Why?
- How we can access “Concrete” methods inside abstract classes?

Summary

- Abstraction
- Abstraction in JAVA
- Abstract Methods
- Abstract Classes
- Hands-On

References

- https://docs.oracle.com/javase/tutorial/java/landl/abstract.html

- How To Program (Early Objects)
  - By H .Deitel and P. Deitel
- Headfirst Java
  - By Kathy Sierra and Bert Bates

Questions ???

Thank You