Benazir Bhutto Shaheed University Lyari, Karachi

Department of Computing Science & Information Technology

Abstraction in Object Oriented Programming

Course: Object Oriented Programming

Defining Abstraction

- Abstraction is the process of extracting common features from specific examples

- Abstraction is a process of defining the essential concepts while ignoring the inessential details

Different Types of Abstraction

- **Data Abstraction**

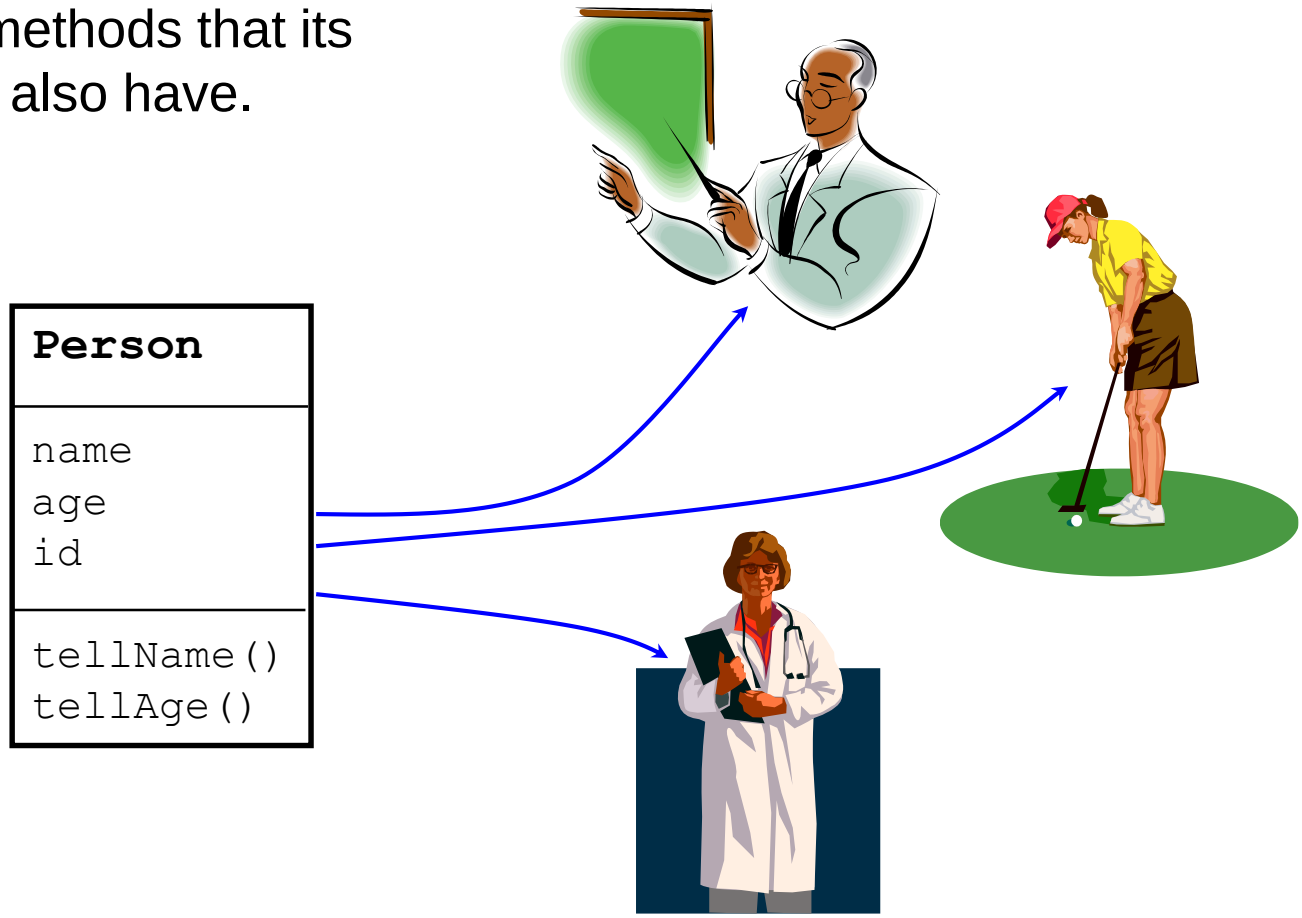
  Programming languages define constructs to simplify the way information is presented to the programmer.
- **Functional Abstraction**

  Programming languages have constructs that ‘gift wrap’ very complex and low level instructions into instructions that are much more readable.
- **Object Abstraction**

  OOP languages take the concept even further and abstract programming constructs as *objects*.

Class as Abstraction

- A class is an abstraction of its instances. It defines all the attributes and methods that its instances must also have.



Points to Remember

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

Encapsulation in Object Oriented Programming

Defining Encapsulation

- ***Encapsulation*** is the process of hiding an object's implementation from another object, while presenting only the interfaces that should be visible.

Principles of Encapsulation

"Don't ask how I do it, but this is what I can do"

- The encapsulated object

"I don't care how, just do your job, and I'll do mine"

- One encapsulated object to another

Encapsulation in Java

- **Encapsulation in Java** is a *process of wrapping code and data together into a single unit*, for example, a capsule which is mixed of several medicines.
- We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

Encapsulating a Class

- Members of a class must always be declared with the minimum level of visibility.
- Provide *setters and getters* (also known as accessors/mutators) to allow *controlled* access to private data.
- Provide other public methods (known as *interfaces*) that other objects must adhere to in order to interact with the object.

Setters and Getters

- Setters and Getters allow controlled access to class data
- *Setters* are methods that (only) alter the state of an object
  - Use setters to validate data before changing the object state
- *Getters* are methods that (only) return information about the state of an object
  - Use getters to format data before returning the object's state

```
private char sex;

public void setSex(char s) {
    // validate here
    sex = s;
}
public char getSex() {
    // format here
    return sex;
}
```

Advantage of Encapsulation in Java

- By providing only a setter or getter method, you can make the class **read-only or write-only**. In other words, you can skip the getter or setter methods.
- It provides you the **control over the data**. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.
- It is a way to achieve **data hiding** in Java because other class will not be able to access the data through the private data members.
- The encapsulate class is **easy to test**. So, it is better for unit testing.

Encapsulation Example

```
public static void main(String[] args) {
    // instantiate several objects
    Person p1 = new Person();
    Person p2 = new Person();
    Person p3 = new Person();

    // access instance variables using setters
    p1.setName("Vincent"); p1.setSex('M');
    p1.setAge(8);
    p2.setName("Janice"); p2.setSex('F');
    p1.setAge(19);
    p3.setName("Ricky"); p3.setSex('M');
    p3.setAge(34);

    // access static variables directly
    Person.maleCount=2;
    Person.femaleCount=1;

    // access instance methods
    p1.tellSex(); p1.tellAge();
    p2.tellSex(); p2.tellAge();
    p3.tellSex(); p3.tellAge();

    // access static method
    Person.showSexDistribution();
}
```

```
class Person {
    // set variables to private
    private static int maleCount;
    private static int femaleCount;
    private String name;
    private char sex;
    private int age;

    /*
     * setters & getters, set to public
     */
    public int getAge() { return age;}
    public void setAge(int a) { age = a;}
    public String getName() { return name;}
    public void setName(String n) { name = n;}
    public char getSex() { return sex;}
    public void setSex(char s) { sex = s;}
    /*
     * set other methods as interfaces
     */
    public static void showSexDistribution() {
          // implementation here
    }
    public void tellSex() {

    }
    publi

    }
}
```

```
I'm Male.
I'm just a kid.
I'm Female.
I'm a teenager.
I'm Male.
I'm a grown up.
Majority are male.
```

Key Points

- *Abstraction* is the process of formulating general concepts by extracting common properties of instances.
- A class is an abstraction of its instances.
- A Java Class denotes a category of objects.
- Class members refer to its *fields* and *methods.*
- *Static members* are variables and methods belonging to a class.
- *Instance members* are variables and methods belonging to objects.
- Instantiating a class means creating objects of its own type.
- Class modifiers include: (no modifier), `public`, `abstract`, `final` and `strictfp`.
- Member modifiers include: (no modifier), `public`, `protected`, `private`, `static`, `final`, `abstract`, `strictfp`, `synchronized`, `native`, `transient` and `volatile`.

Key Points (Continued)

- Encapsulation hides implementation details of a class.
- Encapsulating a class means declaring members with minimum level of visibility.
- *Setters* are methods whose only function is to alter the state of an object in a controlled manner.
- *Getters* are methods which only function is to return information about the state of an object.
- *Constructors* are methods which set the initial state of an object upon creation of the object.

Constructors

- *Constructors* are methods which set the initial state of an object
- Constructors are called when an object is created using the `new` operator
- A *default constructor* is a constructor with no parameters, it initializes the instance variables to default values
- Restrictions on constructors
  - constructor name must be the same as the class name
  - constructor cannot return a value, not even `void`
  - only an access modifier is allowed