OBJECT ORIENTED PROGRAMMING

Affefah Qureshi

Department of Computer Science

Iqra University, Islamabad Campus.

CONTROLLING ACCESS TO MEMBERS

- Access Modifiers
  - Private:
    - The private access modifier restricts access to the member within the class where it is declared. It cannot be accessed from outside the class. Private members are only visible within the class itself.
  - Public
    - The public access modifier allows the member to be accessed from anywhere, both within the class and outside of it. Public members have the widest scope among all other modifiers and can be accessed from any class in the program
  - Protected (will study later)

EXAMPLE

```
public class Time1{
        private int hour;
        private int minute;
        private int second;

public void setTime(int h, int m, int s){
        if((h>=0 && h<24) && (m>=0 && m<60) && (s>=0 && s<60)) {
                hour = h;
                minute = m;
                second = s; }
        else
                system.out.println("Invalid Time");
}}
```

CONTROLLING ACCESS TO MEMBERS

- The primary purpose of public methods is to present to the class's clients a view of the services the class provides (the class's public interface).
- Clients need not be concerned with how the class accomplishes its tasks.
- For this reason, the class's private variables and private methods (i.e., its implementation details) are not accessible to its clients.

EXAMPLE

```
public class MemberAccessTest {
    public static void main( String[] args ) {
                        Time1 time = new Time1(); // create and initialize
    Time1 object
    time.hour = 7; // error: hour has private access in Time1
    time.minute = 15; // error: minute has private access in Time1
    time.second = 30; // error: second has private access in Time1
    }}
```

REFERRING THE CURRENT OBJECT'S MEMBER WITH THIS REFERENCE

- Every object can access a reference to itself with keyword this (sometimes called the this reference).
- Use to differentiate between class variables and method parameters.

EXAMPLE

```
public class SimpleTime{
        private int hour;
        private int minute;
        private int second;

public SimpleTime(int hour, int minute, int second){
                this.hour = hour;
                 this. minute = minute;
                 this. second = second; }
}
```

SET AND GET METHOD

- Setter Method:
  - The setter method is used to set or update the value of a private variable in a class. It typically starts with the word "set" followed by the variable name. Setter methods help maintain data integrity and encapsulation by allowing controlled modification of class fields
- Getter Method:
  - The getter method returns the value of a private variable. It starts with the word "get" followed by the variable name. Getter methods provide controlled access to retrieve the value of a class field, ensuring data security and encapsulation

EXAMPLE

```
public class Person {
    private int age;

    // Setter method
    public void setAge(int age) {
        this.age = age;
    }

    // Getter method
    public int getAge() {
        return age;
    }
```

```
public static void main(String[] args) {
        Person person = new Person();

        // Using the setter method to set the
age
        person.setAge(30);

        // Using the getter method to
retrieve the age
        int personAge = person.getAge();
        System.out.println("Person's age: "
+ personAge);
    }
}
```

PRACTICE

- Write a program in Java with a class named Student containing set and get methods, along with name and roll number variables. In the main method, take user input, pass it to the appropriate functions, and then call the get method to display 'Hey [name], your roll number is [roll number].