جامعة الأميرة
نورة بنت عبدالرحمن
كلية علوم الحاسب والمعلومات

CS 111 - Lab

Programming Language 2

Computer Science Department

2026

IQ ACADEMY

Lab 1.1

Objects and Classes

Lab Objectives:

In this lab, a student will practice

• In this lab, the student will practice: • Creating a class declaration. • Declaring instance variables. • Declaring a constructor. • Overloading constructor. • Reading part of a UML class diagram/access modifier.	• Declaring accessors and mutators methods. • Writing a test application to demonstrate the capabilities of another class.

Lab Exercise 1:

Based on the UML class diagram below, create a Java class named **MobilePhone**.
The class should include all required **instance variables**, **constructors**, **setters**, and **getters**. Then, write a **test application** to demonstrate the class capabilities.

MobilePhone
- brand: String - model : String - batteryLevel : int - isOn : boolean
+ <<full constructor>> + MobilePhone(brand : String, model : String) + MobilePhone(brand : String, model : String, batteryLevel : int) + setBrand(brand : String) : void + getBrand() : String + setMode(model : String): void + getModel() : String + setBatteryLevel(batteryLevel : int): void + getBatteryLevel() : int + powerOn() : void + powerOff() : void + useBattery(minutes: int): void + displayInfo() : void

Consider the following points:

a) Implement the constructors shown in the UML using **constructor overloading**.
   • Set batteryLevel to 100 and isOn to false as default values.

b) In setBatteryLevel, validate that the value is between **0 and 100** before assigning it.

c) Implement the following methods:

- **powerOn()**
  - If the phone is already on, display: "Phone is already on". ○ If the battery level is **0**, display: "Battery empty. Cannot turn on.". ○ Otherwise, set isOn to true.
- **powerOff()**: sets isOn to false.
- **useBattery(int minutes)**
  - If the phone is **off**, display: "Phone is off" and do nothing. ○ If minutes is less than or equal to **0**, display: "Invalid usage time". ○ Each minute reduces the battery level by **1**.
  - If the battery level becomes **0 or less**, set it to 0, turn the phone off, and display: "Battery drained. Phone turned off.".
- **displayInfo()**: displays all phone information.

```
public class MobilePhone {
    private String brand;
    private String model;
    private int batteryLevel;
```

```
private boolean isOn;

public MobilePhone(String brand, String model, int batteryLevel, boolean isOn) {
    this.brand = brand;
    this.model = model;
    this.batteryLevel = batteryLevel;
    this.isOn = isOn;
}


public MobilePhone(String brand, String model) {
    this.brand = brand;
    this.model = model;
    this.batteryLevel = 100;
    this.isOn = false;
}

public MobilePhone(String brand, String model, int batteryLevel) {
    this.brand = brand;
    this.model = model;
    this.batteryLevel = batteryLevel;
    this.isOn = false;
}

public void setBrand(String brand) {
    this.brand = brand;
}

public String getBrand() {
    return brand;
}

public void setModel(String model) {
    this.model = model;
```

```
}

public String getModel() {
    return model;
}

public void setBatteryLevel(int batteryLevel) {
    if (batteryLevel >= 0 && batteryLevel <= 100) {
        this.batteryLevel = batteryLevel;
    }
}

public int getBatteryLevel() {
    return batteryLevel;
}

public void powerOn() {
    if (isOn) {
        System.out.println("Phone is already on");
    } else if (batteryLevel == 0) {
        System.out.println("Battery empty. Cannot turn on.");
    } else {
        isOn = true;
    }
}

public void powerOff() {
    isOn = false;
}

public void useBattery(int minutes) {
    if (!isOn) {
```

```
            System.out.println("Phone is off");
            return;
        }
        if (minutes <= 0) {
            System.out.println("Invalid usage time");
            return;
        }
        batteryLevel -= minutes;
        if (batteryLevel <= 0) {
            batteryLevel = 0;
            isOn = false;
            System.out.println("Battery drained. Phone turned off.");
        }
    }

    public void displayInfo() {
        System.out.println("Brand: " + brand);
        System.out.println("Model: " + model);
        System.out.println("Battery Level: " + batteryLevel + "%");
        System.out.println("Is On: " + isOn);
    }
}
```

d) In the **test class**:

- Create **three MobilePhone objects**, using different constructors.
- Using one object, call displayInfo().
- Call powerOn() for the same object, then call displayInfo() again.
- Call useBattery() using the same object.
- Print the **model** of one of the objects.

```
public class MobilePhoneTest {
    public static void main(String[] args) {
        MobilePhone phone1 = new MobilePhone("BrandA", "ModelX");
```

```
        MobilePhone phone2 = new MobilePhone("BrandB", "ModelY", 80);
        MobilePhone phone3 = new MobilePhone("BrandC", "ModelZ");

        phone1.displayInfo();
        phone1.powerOn();
        phone1.displayInfo();
        phone1.useBattery(30);
        System.out.println("Model of phone2: " + phone2.getModel());
    }
}
```

Lab Exercise 2:

Based on the UML class diagram below, create a Java class named **Hotel**.
The class should include all required **instance variables**, **constructors**, **setters**, and **getters**. Then, write a **test application** to demonstrate the class capabilities.

A hotel has a number of rooms available for booking. Some rooms are **standard** and some are **suite** rooms. **Suite rooms are 30% more expensive** than standard rooms.

Hotel
- rooms : int - standard : int - suite : int - price : double
+ Hotel() + Hotel(int rooms, double price) + setRooms(int rooms) : void + setPrice(double price) : void + getRooms() : int + getPrice() : double + getStandard() : int + getSuite() : int + howManyBooked() : int + bookRoom(int n, String type) : boolean + printInfo() : void

Consider the following points:

A) The class **Hotel** has the following **attributes**: **rooms**: total number of rooms in the hotel **standard**: number of booked standard rooms **suite**: number of booked suite rooms **price**: price per night for a standard room

B) Implement the constructors shown in the UML.
- Initializes standard and suite to zero.

C) **howManyBooked()**
Returns the total number of booked rooms.

D) **bookRoom(int n, String type)**
- n is the number of rooms to book
- type is either "standard" or "suite"

- First, check if enough rooms are available
- If available, increase the corresponding counter and return true
  Otherwise, return false

E) **printInfo()**

Prints all hotel information in an organized format as shown in the sample run. F)

```
public class Hotel {
    private int rooms;
    private int standard;
    private int suite;
    private double price;

    public Hotel() {
        standard = 0;
        suite = 0;
    }

    public Hotel(int rooms, double price) {
        this.rooms = rooms;
        this.price = price;
        standard = 0;
        suite = 0;
    }

    public void setRooms(int rooms) {
        this.rooms = rooms;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public int getRooms() {
        return rooms;
    }

    public double getPrice() {
        return price;
    }

    public int getStandard() {
        return standard;
    }

    public int getSuite() {
        return suite;
    }
```

```
    public int howManyBooked() {
        return standard + suite;
    }

    public boolean bookRoom(int n, String type) {
        int booked = howManyBooked();
        if (booked + n > rooms) {
            return false;
        }
        if (type.equalsIgnoreCase("standard")) {
            standard += n;
            return true;
        } else if (type.equalsIgnoreCase("suite")) {
            suite += n;
            return true;
        }
        return false;
    }

    public void printInfo() {
        System.out.println("***********************************");
        System.out.println("          Hotel Info               ");
        System.out.println("***********************************");
        System.out.println("The hotel has " + rooms + " rooms.");
        System.out.println("Only " + howManyBooked() + " rooms have been
booked.");
        System.out.println(standard + " standard rooms with price = " + price
+ " SR per night");
        System.out.println(suite + " suite rooms with price = " + (price * 1.3) +
" SR per night");
    }
}
```

In the **test class**:

1) Declare an object h of class Hotel.
2) Read from the user:
   - the total number of rooms
   - the standard room price

3) Ask the user how many **standard rooms** they want to book.
   - If booking is successful, display a confirmation message.
   - Otherwise, display:

"Requested number of rooms exceeds availability" 4) Ask the user how many **suite rooms** they want to book.
   - Apply the same availability check and message.

4) Print the hotel information.

```
import java.util.Scanner;

public class HotelTest {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Please enter number of rooms and standard room price: ");
      int rooms = scanner.nextInt();
      double price = scanner.nextDouble();

      Hotel h = new Hotel(rooms, price);

      System.out.print("How many standard rooms would you like to book? ");
      int std = scanner.nextInt();
      if (h.bookRoom(std, "standard")) {
         System.out.println(std + " standard rooms have been booked");
      } else {
         System.out.println("Requested number of rooms exceeds availability");
      }

      System.out.print("How many suite rooms would you like to book? ");
      int suite = scanner.nextInt();
      if (h.bookRoom(suite, "suite")) {
         System.out.println(suite + " suite rooms have been booked");
      } else {
         System.out.println("Requested number of rooms exceeds availability");
      }

      h.printInfo();
      scanner.close();
   }
}
```

SAMPLE RUN 1

```
Please enter number of rooms and standard room price: 80 500
How many standard rooms would you like to book? 30
30 standard rooms have been booked
How many suite rooms would you like to book? 20
20 suite rooms have been booked

**********Hotel Info ********** The
hotel has 80 rooms.
Only 50 rooms have been booked.
30 standard rooms with price = 500.0 SR per night
20 suite rooms with price = 650.0 SR per night
```

SAMPLE RUN 2

```
Please enter number of rooms and standard room price: 60 400
How many standard rooms would you like to book? 40
40 standard rooms have been booked
How many suite rooms would you like to book? 30
Requested number of rooms exceeds availability

**********Hotel Info ********** The
hotel has 60 rooms.
Only 40 rooms have been booked.
40 standard rooms with price = 400.0 SR per night
And 0 suite rooms with price = 520.0 SR per night
```