Exercise 1: Answer the following questions **(5)**
1-How is the **final** keyword utilized in the context of inheritance in Java?
2-What purpose do constructors serve in Java, and what different forms can they take?
3-Explain the concept of inheritance in OOP and its practical benefits?
4-What are the key concepts of OOP?
5-How are class (static) variables different from instance variables?

Exercise 2: What is the output of the following Java code? **(3)**

```
1.Person v = new Person("Salim");
2. Person w = v;
3. w.setName("Fatah");
4. Person x = new Person ();
5. x.setName(v.getName());
6. System.out.println(v.getName()+","+w.getName()+","+x.getName());
7. if (w == x) {  w.setName("Sami"); } else { x.setName("Rami"); }
8. System.out.println(v.getName()+","+w.getName()+","+x.getName());
9. v.setName("Safi");
10. if (w != v) { v.setName("Fati") ;} else { w.setName("Ramzi"); }
11. System.out.println(v.getName()+","+w.getName()+","+x.getName());
```

Exercise 3: (12)

You need to design a Hotel Management System with the following classes and functionality:
A-Room Class (Abstract): This class will have the following protected attributes:
roomNumber (int), **pricePerNight** (double), **isOccupied**(Boolean),**floorNumber** (int), **amenities** (An array of Strings (size 3) to hold the amenities provided in the room ("Desk", "TV", , "AirConditioning").The Room class implement:
1-A constructor that initializes all attributes with interactive input for the amenities (up to 3) using class Scanner.
2-Another constructor that initializes attributes directly with validation rule: If roomNumber, pricePerNight, or floorNumber is negative, automatically set it to 0 and print: "Invalid [field name]. Setting to 0." and predefined amenities passed as an array.
3-bookRoom(): Marks the room as booked by setting isOccupied to true .
4-vacateRoom(): Marks the room as vacant by setting isOccupied to false.
5-isAvailable(): Returns true if the room is not occupied, false otherwise.
6-calculateBill(int nights): Calculates the total bill based on the room price per night and applicable service charges for amenities ('Desk': +5,′ TV′ :+10, 'AirConditioning': +20 per night)."
7-abstract method getRoomType(): returns type room (eg : "single room" or "double room") ;

B-SingleRoom Class (Subclass of Room): add a specific attribute:
hasDesk: A boolean indicating whether the room has a desk. The SingleRoom class implement **:**
8-A constructor to initialize the room number, price per night, floor number, and whether the room has a desk.
9-A method getRoomType() that returns "Single".

C-DoubleRoom Class (Subclass of Room): add a specific attribute:
numberOfBeds: An integer indicating the number of beds in the room. The DoubleRoom Class implement:
10-A constructor to initialize the room number, price per night, floor number, and number of beds.
11-A method getRoomType() that returns "Double".

D-Customer Class: This class will manage customer information and their room reservations.
The class should have the following attributes: name (String), id (string), bookedRoom: A reference to the Room object representing the room reserved by the customer. The Customer Class implement:
12-A constructor to initialize the name and the id
13-makeReservation(Room room): Books the given room if it is available.
14-cancelReservation(): Cancels the current room reservation.
15-getReservationDetails(): Displays the reservation details of the customer, such as room number, room type, and price per night.

E-Hotel Class: define four attribute: an array of 10 room, an array of 10 customer, roomcount (int), customercount (int). The Hotel Class implement:
16-addRoom(Room room): Adds a room to the hotel.
17-addCustomer(Customer customer): Adds a customer to the hotel.
18-findAvailableRoom(String type): Returns the first available room of the specified type (e.g., Single or Double).
19-getCheapestAvailableRoom(): Returns the cheapest available room in the hotel based on the price per night.
20-printRoomStatus(): Displays the current status of all rooms (whether they are available or occupied).

F-Test Class
21- create an object of hotel class
22-create and add rooms to the hotel: SingleRoom(101, 100.0, 1, true); SingleRoom(102, 120.0, 1, false); DoubleRoom(201, 150.0, 2, 2); DoubleRoom(202, 180.0, 2, 2);
23-Create customer objects (Alice with ID "C001" and Bob with ID "C002") and add them to the hotel.
24-Make reservations: Alice makes reservation for room 101 and Bob make reservation for room 201.
25- Print the status of rooms before and after a cancellation.

Indications about using Scanner class:

```
Scanner scanner = new Scanner(System.in); // Scanner object for reading user input
scanner.nextFloat(); // Used for reading float values, not strings.
scanner.nextLine(); // Used for reading the whole line (useful for multi-word input).
scanner.nextInt(); // Used for reading integer values, not strings.
equalsIgnoreCase () : method of String class used to compare two strings for equality while ignoring their case differences
```

Important: In Exercise 3, you must strictly use all attribute and class names as defined.