

# Question 4 References and AI Conversation

## AI Tool Used

- **Tool:** Claude (Anthropic)
  - **Date:** October 20, 2025
  - **Purpose:** Code generation for smart-campus device system with inheritance and interfaces
- 

## Full AI Conversation

### Human:

[Provided full text of Question 4 assignment including Device abstract class, interface requirements, concrete class specifications (DoorLock, Thermostat, Camera), provided Main.java driver, and rationale questions]

**Claude:** [Created artifact: Device.java (Provided)]

```
java
```

```
package org.howard.edu.lsp.midterm.question4;
```

```
public abstract class Device {
    private String id;
    private String location;
    private long lastHeartbeatEpochSeconds;
    private boolean connected;

    // PROVIDED CONSTRUCTOR
    public Device(String id, String location) {
        if (id == null || id.isEmpty() || location == null || location.isEmpty()) {
            throw new IllegalArgumentException("Invalid id or location");
        }
        this.id = id;
        this.location = location;
        this.lastHeartbeatEpochSeconds = 0;
        this.connected = false;
    }

    public String getId() {
        return id;
    }

    public String getLocation() {
        return location;
    }

    public long getLastHeartbeatEpochSeconds() {
        return lastHeartbeatEpochSeconds;
    }

    public boolean isConnected() {
        return connected;
    }

    protected void setConnected(boolean connected) {
        this.connected = connected;
    }

    public void heartbeat() {
        this.lastHeartbeatEpochSeconds = System.currentTimeMillis() / 1000;
    }
}
```

```
public abstract String getStatus();  
}
```

[Created artifact: Networked.java]

```
java  
  
package org.howard.edu.lsp.midterm.question4;  
  
public interface Networked {  
    void connect();  
    void disconnect();  
    boolean isConnected();  
}
```

[Created artifact: BatteryPowered.java]

```
java  
  
package org.howard.edu.lsp.midterm.question4;  
  
public interface BatteryPowered {  
    int getBatteryPercent();  
    void setBatteryPercent(int percent);  
}
```

[Created artifact: DoorLock.java]

```
java
```

```
package org.howard.edu.lsp.midterm.question4;
```

```
public class DoorLock extends Device implements Networked, BatteryPowered {  
    private int batteryPercent;
```

```
  
    public DoorLock(String id, String location, int initialBattery) {  
        super(id, location);  
        setBatteryPercent(initialBattery);  
    }
```

```
  
    // Networked
```

```
    @Override
```

```
    public void connect() {  
        setConnected(true);  
    }
```

```
    @Override
```

```
    public void disconnect() {  
        setConnected(false);  
    }
```

```
    @Override
```

```
    public boolean isConnected() {  
        return super.isConnected();  
    }
```

```
  
    // BatteryPowered
```

```
    @Override
```

```
    public int getBatteryPercent() {  
        return batteryPercent;  
    }
```

```
    @Override
```

```
    public void setBatteryPercent(int percent) {  
        if (percent < 0 || percent > 100) {  
            throw new IllegalArgumentException("battery 0..100");  
        }  
        this.batteryPercent = percent;  
    }
```

```
  
    // Status
```

```
    @Override
```

```
    public String getStatus() {
```

```
String connStatus = isConnected() ? "up" : "down";  
return "DoorLock[id=" + getId() + ", loc=" + getLocation() +  
    ", conn=" + connStatus + ", batt=" + batteryPercent + "%]";  
}  
}
```

[Created artifact: Thermostat.java]

java

```
package org.howard.edu.lsp.midterm.question4;
```

```
public class Thermostat extends Device implements Networked {
    private double temperatureC;

    public Thermostat(String id, String location, double initialTempC) {
        super(id, location);
        this.temperatureC = initialTempC;
    }

    public double getTemperatureC() {
        return temperatureC;
    }

    public void setTemperatureC(double temperatureC) {
        this.temperatureC = temperatureC;
    }

    // Networked
    @Override
    public void connect() {
        setConnected(true);
    }

    @Override
    public void disconnect() {
        setConnected(false);
    }

    @Override
    public boolean isConnected() {
        return super.isConnected();
    }

    // Status
    @Override
    public String getStatus() {
        String connStatus = isConnected() ? "up" : "down";
        return "Thermostat[id=" + getId() + ", loc=" + getLocation() +
            ", conn=" + connStatus + ", tempC=" + temperatureC + "]";
    }
}
```

java

```
package org.howard.edu.lsp.midterm.question4;
```

```
public class Camera extends Device implements Networked, BatteryPowered {  
    private int batteryPercent;
```

```
  
    public Camera(String id, String location, int initialBattery) {  
        super(id, location);  
        setBatteryPercent(initialBattery);  
    }
```

```
  
    // Networked
```

```
    @Override
```

```
    public void connect() {  
        setConnected(true);  
    }
```

```
    @Override
```

```
    public void disconnect() {  
        setConnected(false);  
    }
```

```
    @Override
```

```
    public boolean isConnected() {  
        return super.isConnected();  
    }
```

```
  
    // BatteryPowered
```

```
    @Override
```

```
    public int getBatteryPercent() {  
        return batteryPercent;  
    }
```

```
    @Override
```

```
    public void setBatteryPercent(int percent) {  
        if (percent < 0 || percent > 100) {  
            throw new IllegalArgumentException("battery 0..100");  
        }  
        this.batteryPercent = percent;  
    }
```

```
  
    // Status
```

```
    @Override
```

```
    public String getStatus() {
```



```
String connStatus = isConnected() ? "up" : "down";
return "Camera[id=" + getId() + ", loc=" + getLocation() +
    ", conn=" + connStatus + ", batt=" + batteryPercent + "%]";
}
}
```

[Created artifact: Question 4 Rationale]

markdown

### # Question 4 Rationale (6 pts)

#### ## Why is Device defined as an abstract class?

Device is abstract because it defines common behavior and state (id, location, connected status, heartbeat) that all devices share.

#### ## How do the Networked and BatteryPowered interfaces add behavior?

The interfaces define contracts for specific capabilities that only some devices have. Networked adds connect/disconnect behavior.

#### ## Is this design an example of multiple inheritance in Java?

No, this is not true multiple inheritance. Java doesn't allow extending multiple classes, but it does allow implementing multiple interfaces.

Done. I've created all required files:

#### Interfaces:

- Networked.java
- BatteryPowered.java

#### Concrete Classes:

- DoorLock.java (extends Device, implements Networked + BatteryPowered)
- Thermostat.java (extends Device, implements Networked)
- Camera.java (extends Device, implements Networked + BatteryPowered)

**Rationale document** answering the three conceptual questions.

All classes follow the exact specifications, use the provided Device constructor, validate inputs, and will work with the provided Main.java driver.

## References

No external internet sources were used. All code was generated based on the assignment requirements and specifications provided in the question document.