# Sample Deliverable A (Week 1)

## Domain & OOP Design

**Chosen Domain**: **Local Craft Marketplace**

- Rationale: A Gaborone-based online platform for selling handmade crafts.
- We decided on two main domain classes so far: **`Product`** and **`Seller`**.

### 1. Minimal Domain Classes

```
// Product.java
package bw.crafts;

public class Product {
    private String productName;
    private double price;
    private Seller seller;  // association with a Seller object

    public Product(String productName, double price, Seller seller) {
        this.productName = productName;
        this.price = price;
        this.seller = seller;
    }

    // getters & setters
    public String getProductName() { return productName; }
    public void setProductName(String productName) { this.productName =
productName; }

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

    public Seller getSeller() { return seller; }
    public void setSeller(Seller seller) { this.seller = seller; }
}

// Seller.java
package bw.crafts;

public class Seller {
    private String sellerName;
    private String location; // e.g. "Gaborone", "Francistown"

    public Seller(String sellerName, String location) {
        this.sellerName = sellerName;
        this.location = location;
    }

    // getters & setters
    public String getSellerName() { return sellerName; }
    public void setSellerName(String sellerName) { this.sellerName =
sellerName; }

    public String getLocation() { return location; }
    public void setLocation(String location) { this.location = location; }
}
```

- **Encapsulation**: Fields are private, with basic getters/setters.
- **Polymorphism**: We plan to add either an **interface** or an **abstract** class in Week 2 (e.g., `Sortable` or a sub-class for different product categories).

### 2. GitHub Setup

- Repository Name: **`local-craft-market-mini-project`**
- Link (example): `https://github.com/CraftMarketplaceTeam/local-craft-market-mini-project`
- Each group member made initial commits:
  - *Commit 1:* "Set up domain classes Product, Seller."
  - *Commit 2:* "Added minimal getters/setters."

### 3. Next Steps (End of Week 1)

- Next week, we'll **add sorting** methods (selection, insertion) in a `Sorter` utility class.
- Then we'll create a **custom exception** for negative prices.
- We plan to do a **linear search** for product name, a **binary search** for price (iterative).

**Short Reflection**: We have the domain classes established, a Git repo, and minimal structure. Next, we handle sorting & searching.

---

# Sample Deliverable B (Week 1)

## Domain & OOP Design

**Chosen Domain**: **University Student Roster**

- Rationale: Track university students, focusing on `Student` and possibly `GraduateStudent` to demonstrate **inheritance**.

### 1. Minimal Domain Classes

```
// Student.java
package bw.uni;

public class Student {
    private String name;
    private int score;        // e.g., exam or overall score
    private String faculty;

    public Student(String name, int score, String faculty) {
        this.name = name;
        this.score = score;
        this.faculty = faculty;
    }

    // getters & setters
    public String getName() { return name; }
```

```
    public void setName(String name) { this.name = name; }

    public int getScore() { return score; }
    public void setScore(int score) { this.score = score; }

    public String getFaculty() { return faculty; }
    public void setFaculty(String faculty) { this.faculty = faculty; }
}

// GraduateStudent.java
package bw.uni;

public class GraduateStudent extends Student {
    private String thesisTitle;

    public GraduateStudent(String name, int score, String faculty, String
thesisTitle) {
        super(name, score, faculty);
        this.thesisTitle = thesisTitle;
    }

    public String getThesisTitle() { return thesisTitle; }
    public void setThesisTitle(String thesisTitle) { this.thesisTitle =
thesisTitle; }
}
```

- **Encapsulation**: Private fields with getters/setters.
- **Inheritance**: `GraduateStudent` extends `Student`.
- Polymorphism will be relevant if we treat a `Student` array that also contains `GraduateStudent`.

## 2. GitLab Setup

- Repository Name: **`university-roster-csi142`**
- Link (example): `https://gitlab.com/UniRosterGroup/university-roster-csi142`
- Current commits:
    - *Commit 1:* "Created Student.java, GraduateStudent.java classes."
    - *Commit 2:* "Added constructor & basic fields, readme placeholders."

## 3. Next Steps (End of Week 1)

- In Week 2, we'll create a **RosterSorter** class to handle:
    - **Selection sort** by `score`.
    - **Insertion sort** by `score`.
    - **Linear search** by `name`.
    - **Binary search** (iterative) by `score`.
- We plan to add a **custom exception** for negative or out-of-range `score`.

**Short Reflection**: We've completed core domain classes and set up version control. Next, we implement sorting/searching in line with the assignment specs.