Project 1

```
import java.util.*;
import java.util.stream.Collectors;

class Patient {
    private String name;
    private int age;
    private String diagnosis;
    private String treatment;

    public Patient(String name, int age, String diagnosis, String treatment) {
        this.name = name;
        this.age = age;
        this.diagnosis = diagnosis;
        this.treatment = treatment;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public String getDiagnosis() {
        return diagnosis;
    }

    public String getTreatment() {
        return treatment;
    }

    @Override
    public String toString() {
        return name + " (" + age + " years old, Diagnosis: " + diagnosis + ", Treatment: " +
treatment + ")";
    }
}

public class HealthcareDataAnalysis {
    public static void main(String[] args) {
        List<Patient> patients = Arrays.asList(
            new Patient("John Doe", 30, "Flu", "Rest"),
            new Patient("Jane Smith", 45, "Diabetes", "Insulin"),
            new Patient("Alice Brown", 60, "Heart Disease", "Medication"),
            new Patient("Bob Johnson", 25, "Flu", "Rest"),
            new Patient("Charlie Davis", 50, "Diabetes", "Diet"),
            new Patient("Eve Wilson", 40, "Heart Disease", "Medication")
        );

        // 1. Filter patients with 'Diabetes'
        System.out.println("Patients with Diabetes:");
        patients.stream()
            .filter(p -> p.getDiagnosis().equalsIgnoreCase("Diabetes"))
            .forEach(System.out::println);

        // 2. Calculate the average age of patients with 'Heart Disease'
        OptionalDouble averageAgeHeartDisease = patients.stream()
            .filter(p -> p.getDiagnosis().equalsIgnoreCase("Heart Disease"))
            .mapToInt(Patient::getAge)
            .average();

        averageAgeHeartDisease.ifPresent(avg -> System.out.println("Average Age of Patients
with Heart Disease: " + avg));

        // 3. Group patients by their treatment type
        System.out.println("Patients Grouped by Treatment:");
        Map<String, List<Patient>> groupedByTreatment = patients.stream()
            .collect(Collectors.groupingBy(Patient::getTreatment));

        groupedByTreatment.forEach((treatment, patientList) -> {
            System.out.println(treatment + ": " + patientList);
        });

        // 4. Find the most common diagnosis
        System.out.println("Most Common Diagnoses:");
        Map<String, Long> diagnosisCount = patients.stream()
            .collect(Collectors.groupingBy(Patient::getDiagnosis, Collectors.counting()));

        diagnosisCount.entrySet().stream()
            .max(Map.Entry.comparingByValue())
            .ifPresent(entry -> System.out.println("Most Common Diagnosis: " + entry.getKey() +
" (Count: " + entry.getValue() + ")"));
    }
}
```

Project 2

```
import java.util.*;
import java.util.stream.Collectors;

class Product {
    private String name;
    private String category;
```

```java
    private int quantity;
    private double price;

    public Product(String name, String category, int quantity, double price) {
        this.name = name;
        this.category = category;
        this.quantity = quantity;
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public String getCategory() {
        return category;
    }

    public int getQuantity() {
        return quantity;
    }

    public double getPrice() {
        return price;
    }

    public double getTotalValue() {
        return quantity * price;
    }

    @Override
    public String toString() {
        return name + " (" + category + "): " + quantity + " units, $" + price;
    }
}

public class ProductInventoryManagement {
    public static void main(String[] args) {
        List<Product> products = Arrays.asList(
            new Product("Laptop", "Electronics", 5, 800),
            new Product("Phone", "Electronics", 10, 500),
            new Product("Desk", "Furniture", 0, 200),
            new Product("Chair", "Furniture", 15, 150),
            new Product("Headphones", "Electronics", 20, 100)
        );

        // 1. Filter out out-of-stock products
        System.out.println("In-stock Products:");
        products.stream()
            .filter(p -> p.getQuantity() > 0)                  // Intermediate operation 1: Filter
            .forEach(System.out::println);

        // 2. Calculate the total value of in-stock products
        double totalInventoryValue = products.stream()
            .filter(p -> p.getQuantity() > 0)                  // Intermediate operation 2: Filter
            .mapToDouble(Product::getTotalValue)                     // Intermediate operation 3:
MapToDouble
            .sum();                                            // Terminal operation

        System.out.println("Total Inventory Value: $" + totalInventoryValue);

        // 3. Group products by category
        System.out.println("Products Grouped by Category:");
        Map<String, List<Product>> productsByCategory = products.stream()
            .filter(p -> p.getQuantity() > 0)                  // Intermediate operation 4: Filter
            .collect(Collectors.groupingBy(Product::getCategory));     // Intermediate operation 5:
Collect

        productsByCategory.forEach((category, productList) -> {
            System.out.println(category + ": " + productList);
        });

        // 4. Sort products by price within each category
        System.out.println("Products Sorted by Price within Each Category:");
        productsByCategory.forEach((category, productList) -> {
            System.out.println(category + ": " +
                productList.stream()
                    .sorted(Comparator.comparingDouble(Product::getPrice)) // Intermediate
operation 6: Sorted
                    .collect(Collectors.toList())
            );
        });
    }
}
```