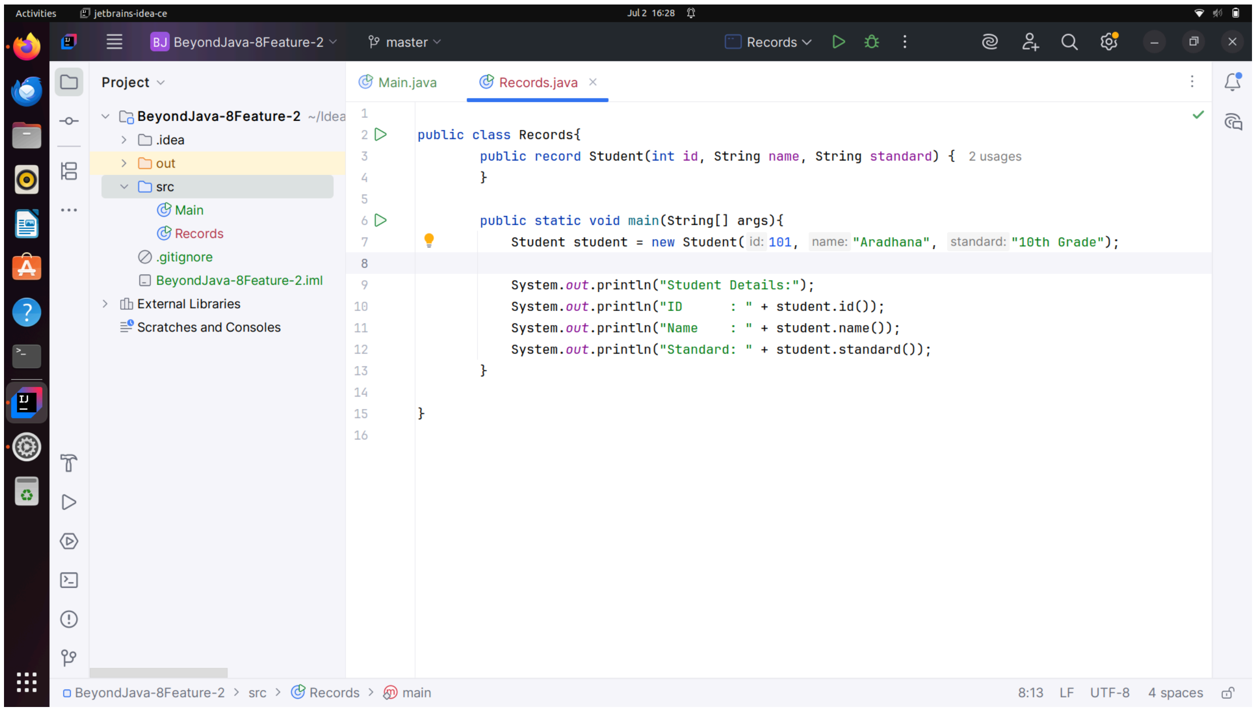
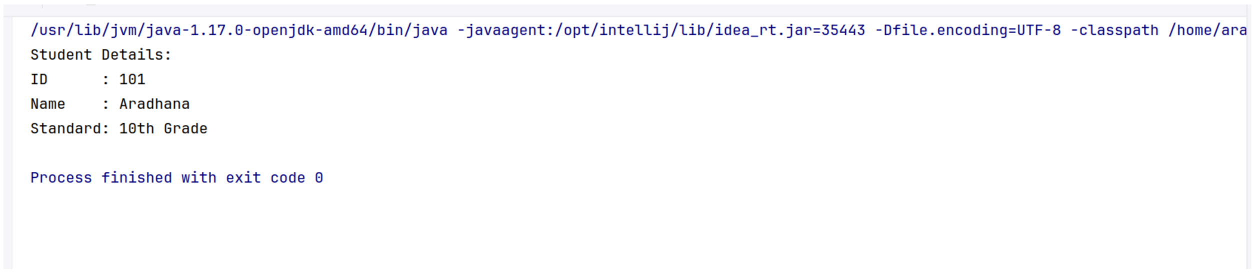
1. Create a Record for the Student with the following Fields: id name standard



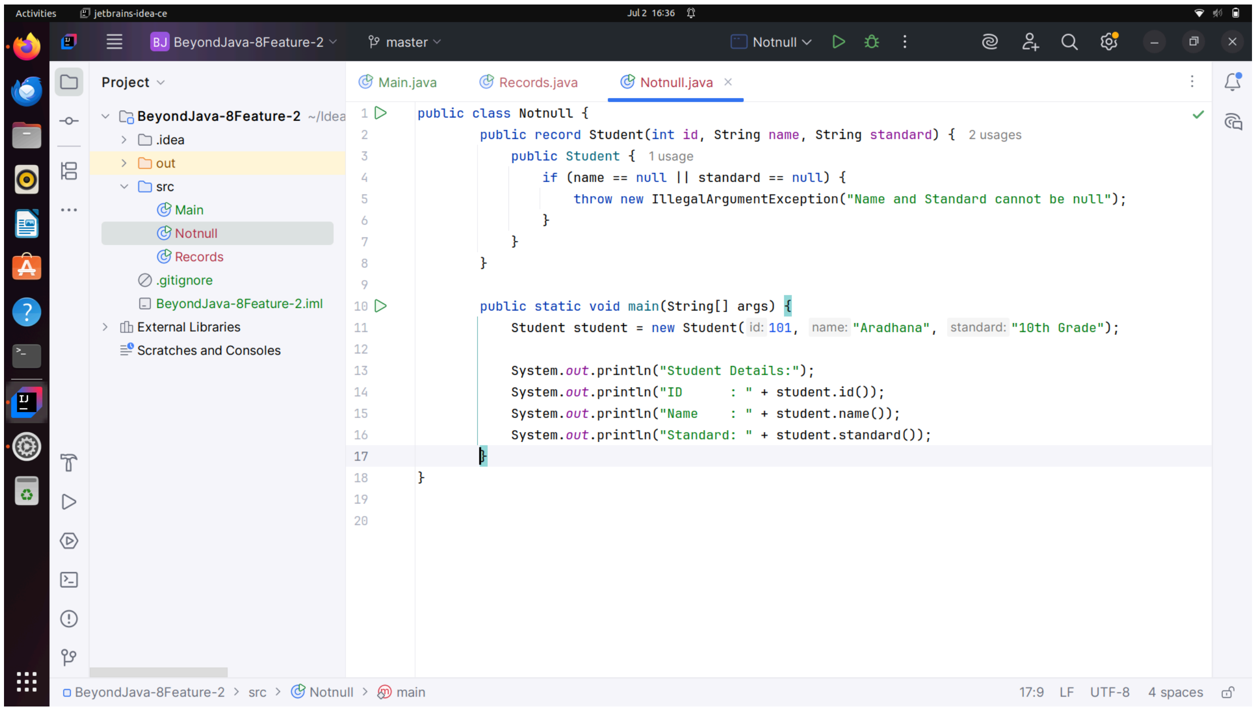
```java
public class Records{
        public record Student(int id, String name, String standard) {
        }

        public static void main(String[] args){
            Student student = new Student( id: 101,  name: "Aradhana",  standard: "10th Grade");

            System.out.println("Student Details:");
            System.out.println("ID      : " + student.id());
            System.out.println("Name    : " + student.name());
            System.out.println("Standard: " + student.standard());
        }

}
```

Output



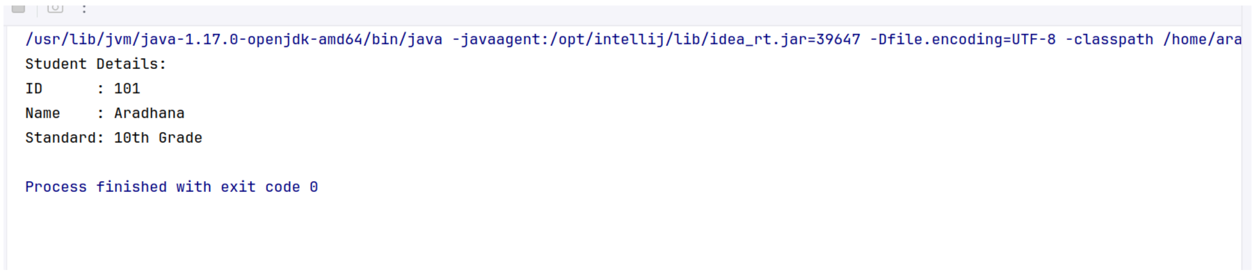
```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=35443 -Dfile.encoding=UTF-8 -classpath /home/ara
Student Details:
ID      : 101
Name    : Aradhana
Standard: 10th Grade

Process finished with exit code 0
```

2. Make sure that no null values should be used for initialization.



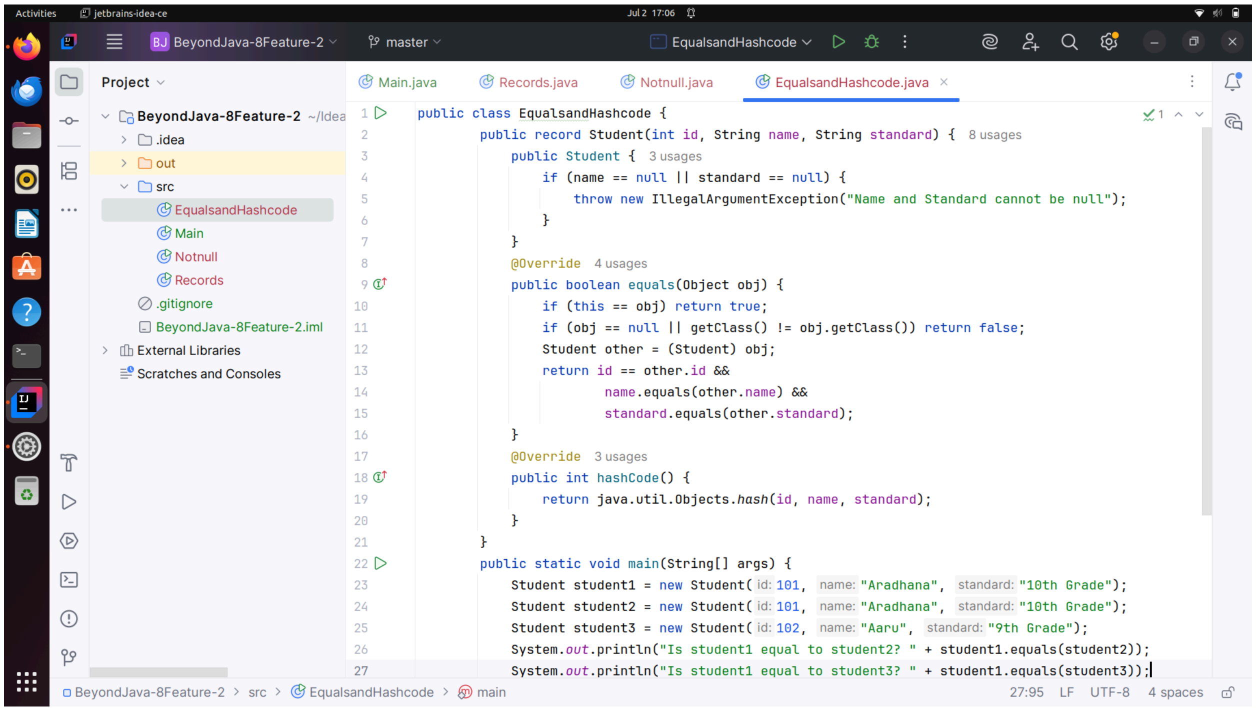
```
public class Notnull {
        public record Student(int id, String name, String standard) {  2 usages
            public Student {  1 usage
                if (name == null || standard == null) {
                    throw new IllegalArgumentException("Name and Standard cannot be null");
                }
            }
        }

        public static void main(String[] args) {
            Student student = new Student( id: 101,  name: "Aradhana",  standard: "10th Grade");

            System.out.println("Student Details:");
            System.out.println("ID       : " + student.id());
            System.out.println("Name     : " + student.name());
            System.out.println("Standard: " + student.standard());
        }
}
```

Output:

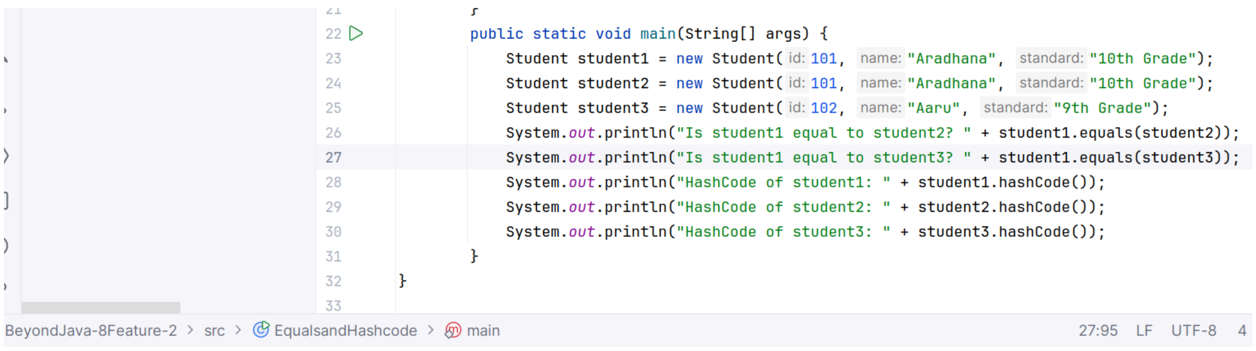


```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=39647 -Dfile.encoding=UTF-8 -classpath /home/ara
Student Details:
ID       : 101
Name     : Aradhana
Standard: 10th Grade

Process finished with exit code 0
```

3. Use equal and hashCode methods with Student records

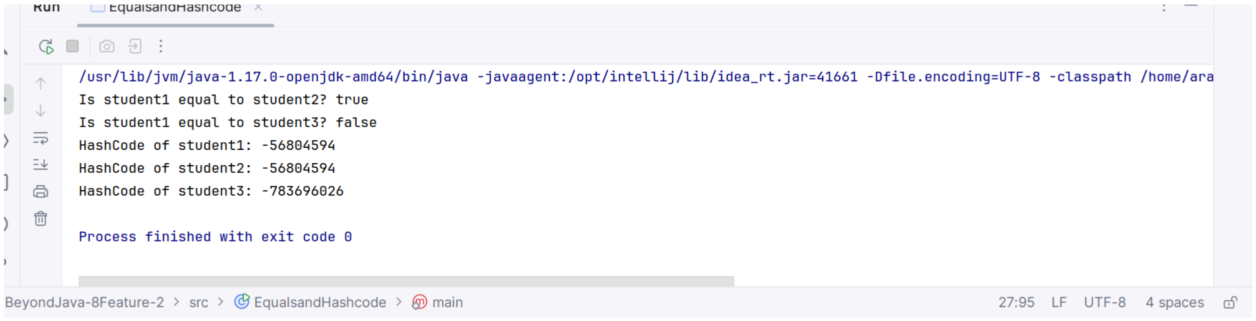


```java
public class EqualsandHashcode {
        public record Student(int id, String name, String standard) {
            public Student {
                if (name == null || standard == null) {
                    throw new IllegalArgumentException("Name and Standard cannot be null");
                }
            }
            @Override
            public boolean equals(Object obj) {
                if (this == obj) return true;
                if (obj == null || getClass() != obj.getClass()) return false;
                Student other = (Student) obj;
                return id == other.id &&
                        name.equals(other.name) &&
                        standard.equals(other.standard);
            }
            @Override
            public int hashCode() {
                return java.util.Objects.hash(id, name, standard);
            }
        }
        public static void main(String[] args) {
            Student student1 = new Student( 101, "Aradhana", "10th Grade");
            Student student2 = new Student( 101, "Aradhana", "10th Grade");
            Student student3 = new Student( 102, "Aaru", "9th Grade");
            System.out.println("Is student1 equal to student2? " + student1.equals(student2));
            System.out.println("Is student1 equal to student3? " + student1.equals(student3));
```



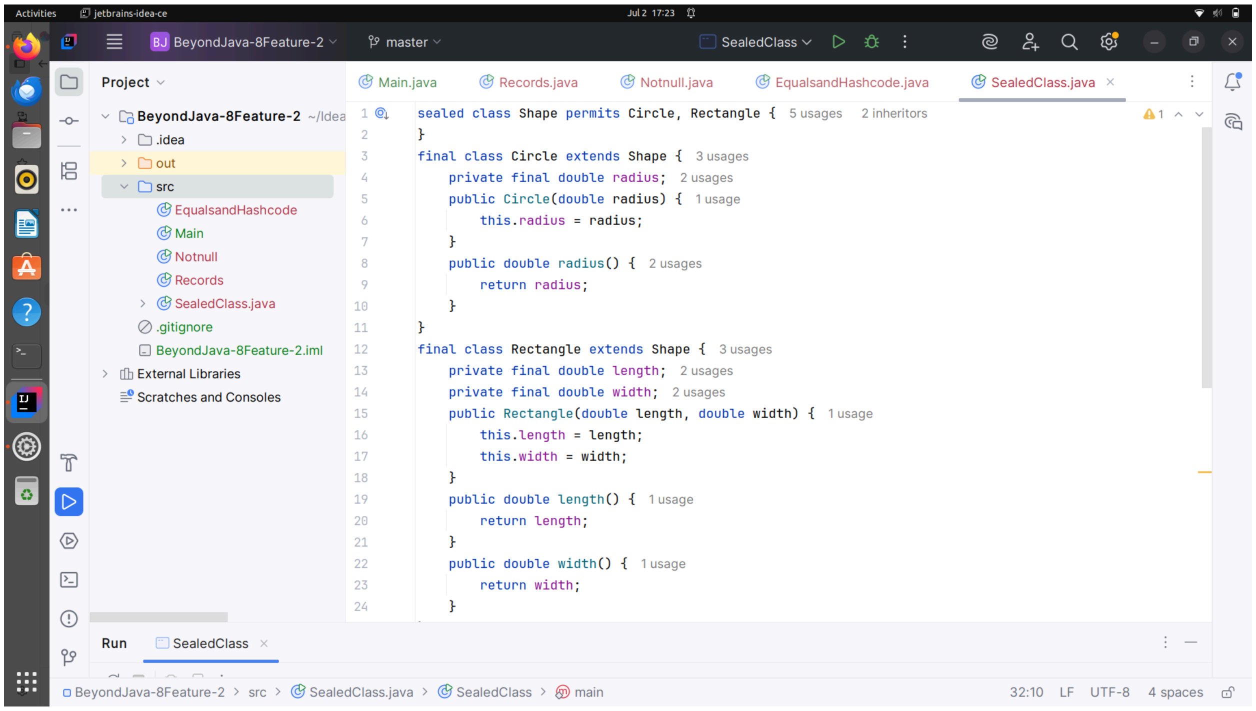
```java
        public static void main(String[] args) {
            Student student1 = new Student( 101, "Aradhana", "10th Grade");
            Student student2 = new Student( 101, "Aradhana", "10th Grade");
            Student student3 = new Student( 102, "Aaru", "9th Grade");
            System.out.println("Is student1 equal to student2? " + student1.equals(student2));
            System.out.println("Is student1 equal to student3? " + student1.equals(student3));
            System.out.println("HashCode of student1: " + student1.hashCode());
            System.out.println("HashCode of student2: " + student2.hashCode());
            System.out.println("HashCode of student3: " + student3.hashCode());
        }
}
```

Output:

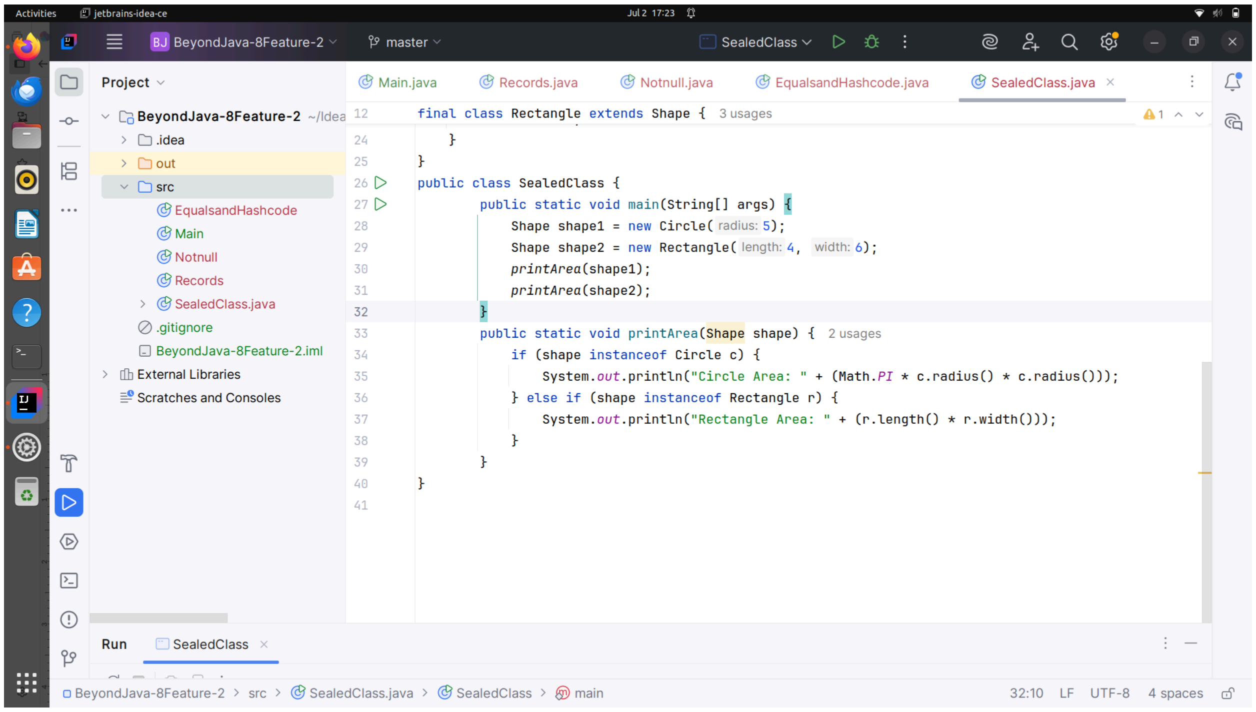


```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=41661 -Dfile.encoding=UTF-8 -classpath /home/ara
Is student1 equal to student2? true
Is student1 equal to student3? false
HashCode of student1: -56804594
HashCode of student2: -56804594
HashCode of student3: -783696026

Process finished with exit code 0
```

4. Use a Sealed class Class concept to create a class hierarchy

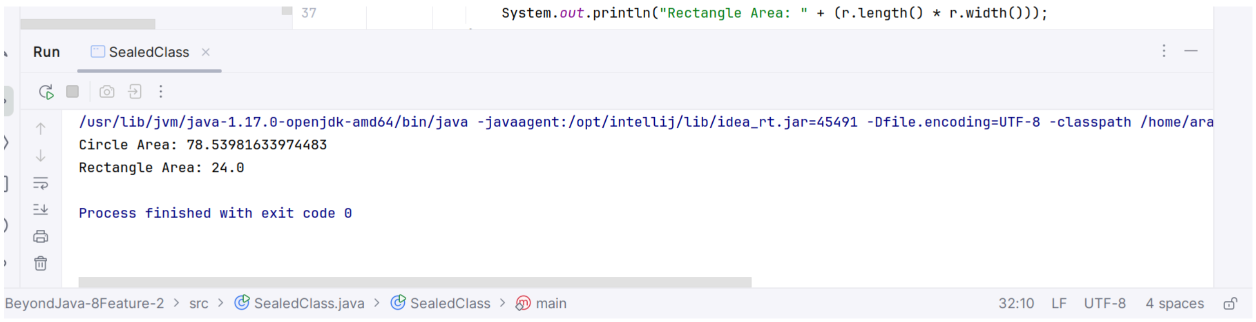


```java
sealed class Shape permits Circle, Rectangle {
}
final class Circle extends Shape {
    private final double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    public double radius() {
        return radius;
    }
}
final class Rectangle extends Shape {
    private final double length;
    private final double width;
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    public double length() {
        return length;
    }
    public double width() {
        return width;
    }
```



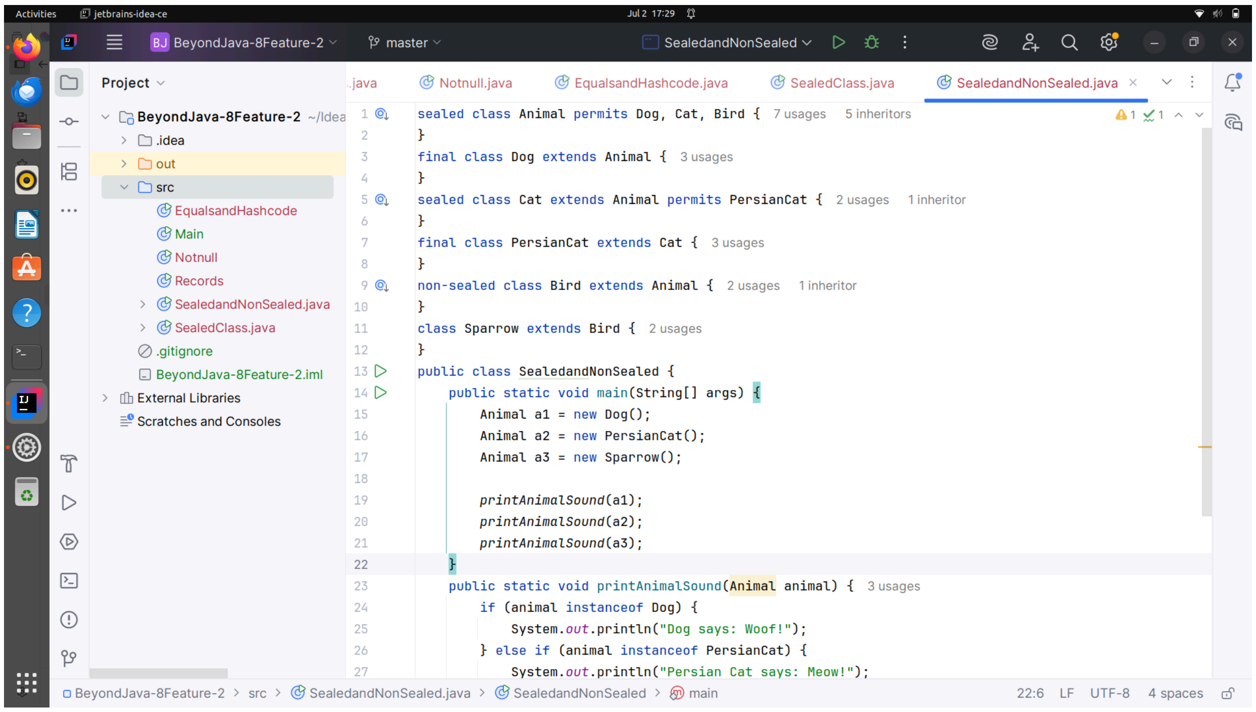
```java
final class Rectangle extends Shape {
    }
}
public class SealedClass {
        public static void main(String[] args) {
            Shape shape1 = new Circle(5);
            Shape shape2 = new Rectangle(4, 6);
            printArea(shape1);
            printArea(shape2);
        }
        public static void printArea(Shape shape) {
            if (shape instanceof Circle c) {
                System.out.println("Circle Area: " + (Math.PI * c.radius() * c.radius()));
            } else if (shape instanceof Rectangle r) {
                System.out.println("Rectangle Area: " + (r.length() * r.width()));
            }
        }
}
```

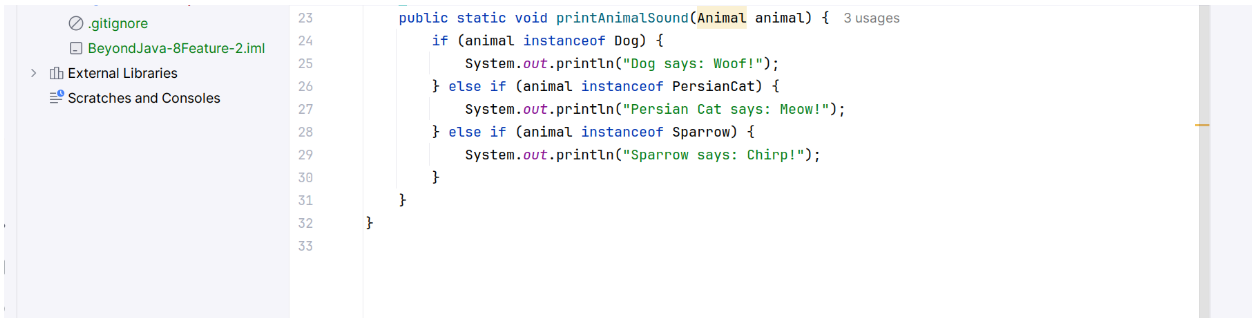
Output:


```
System.out.println("Rectangle Area: " + (r.length() * r.width()));
```

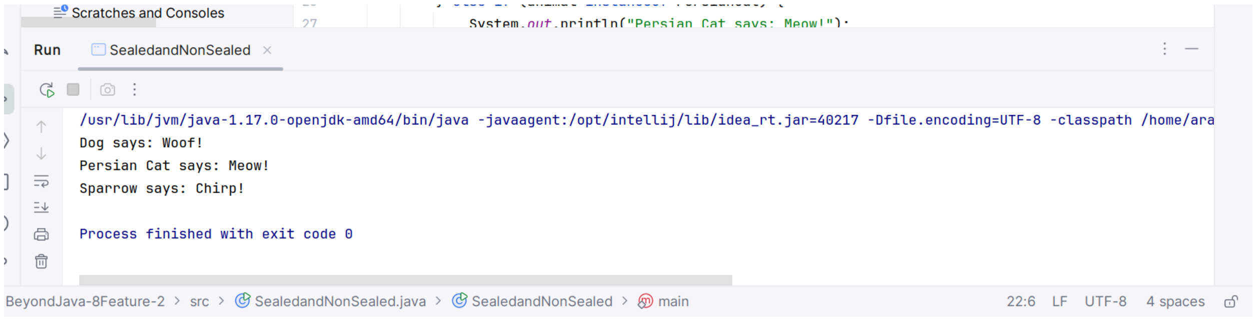
```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=45491 -Dfile.encoding=UTF-8 -classpath /home/ara
Circle Area: 78.53981633974483
Rectangle Area: 24.0

Process finished with exit code 0
```

5. Mark Child classes as final, sealed, and non sealed and observe their behavior


```java
sealed class Animal permits Dog, Cat, Bird {
}
final class Dog extends Animal {
}
sealed class Cat extends Animal permits PersianCat {
}
final class PersianCat extends Cat {
}
non-sealed class Bird extends Animal {
}
class Sparrow extends Bird {
}
public class SealedandNonSealed {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        Animal a2 = new PersianCat();
        Animal a3 = new Sparrow();

        printAnimalSound(a1);
        printAnimalSound(a2);
        printAnimalSound(a3);
    }
    public static void printAnimalSound(Animal animal) {
        if (animal instanceof Dog) {
            System.out.println("Dog says: Woof!");
        } else if (animal instanceof PersianCat) {
            System.out.println("Persian Cat says: Meow!");
```

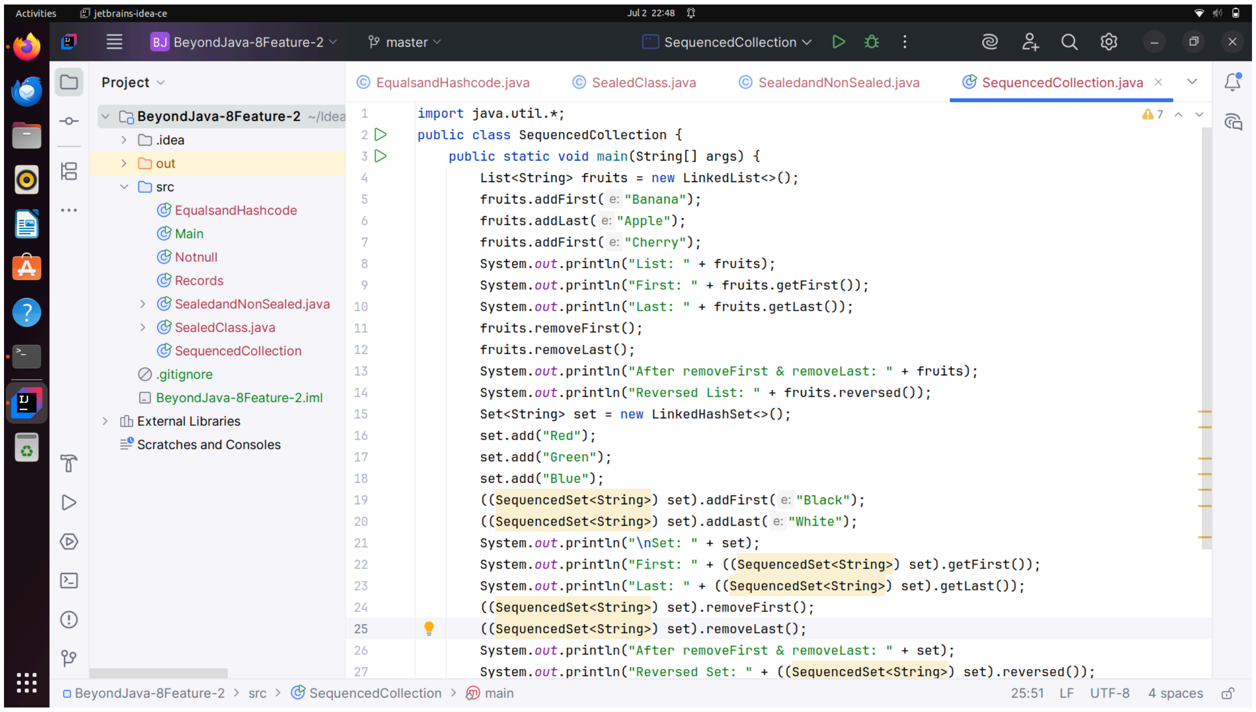

```java
    public static void printAnimalSound(Animal animal) {
        if (animal instanceof Dog) {
            System.out.println("Dog says: Woof!");
        } else if (animal instanceof PersianCat) {
            System.out.println("Persian Cat says: Meow!");
        } else if (animal instanceof Sparrow) {
            System.out.println("Sparrow says: Chirp!");
        }
    }
}
```

Output:



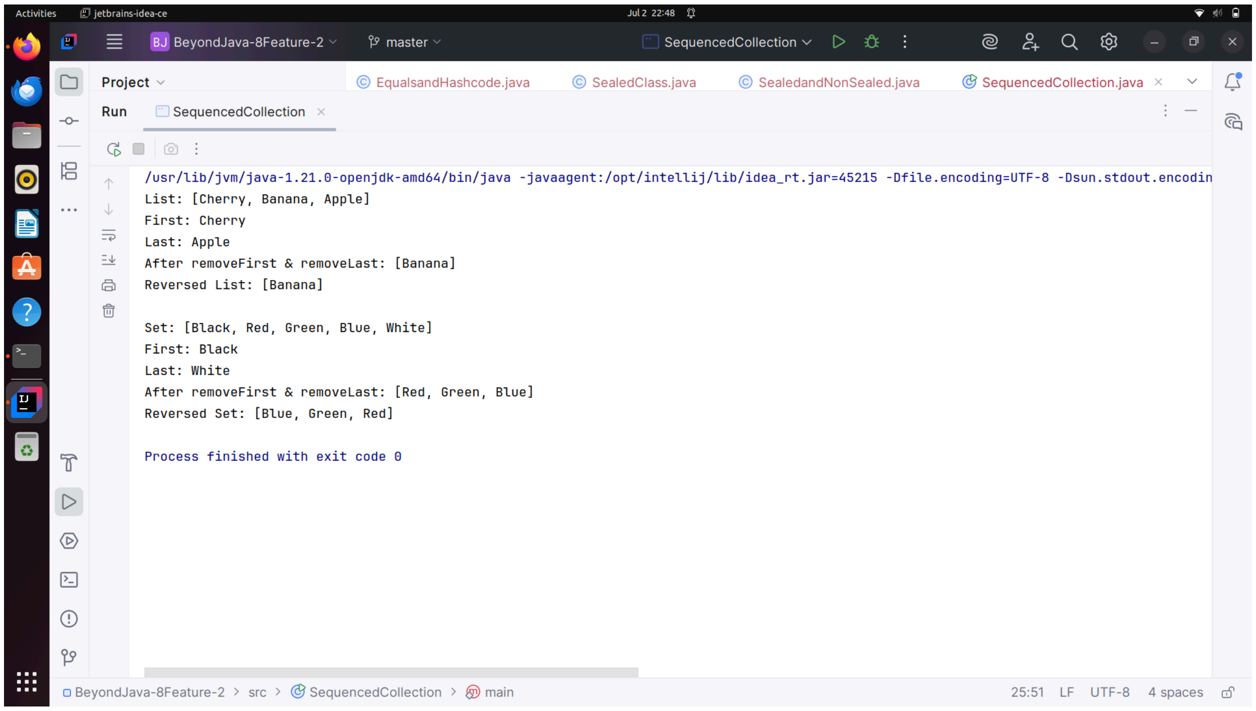
```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=40217 -Dfile.encoding=UTF-8 -classpath /home/ara
Dog says: Woof!
Persian Cat says: Meow!
Sparrow says: Chirp!

Process finished with exit code 0
```

6. Demonstrate the use of addFirst(), addLast, removeFirst(), removeLast, getFirst(), getLast(), reversed() in Set and List Sequenced collections.



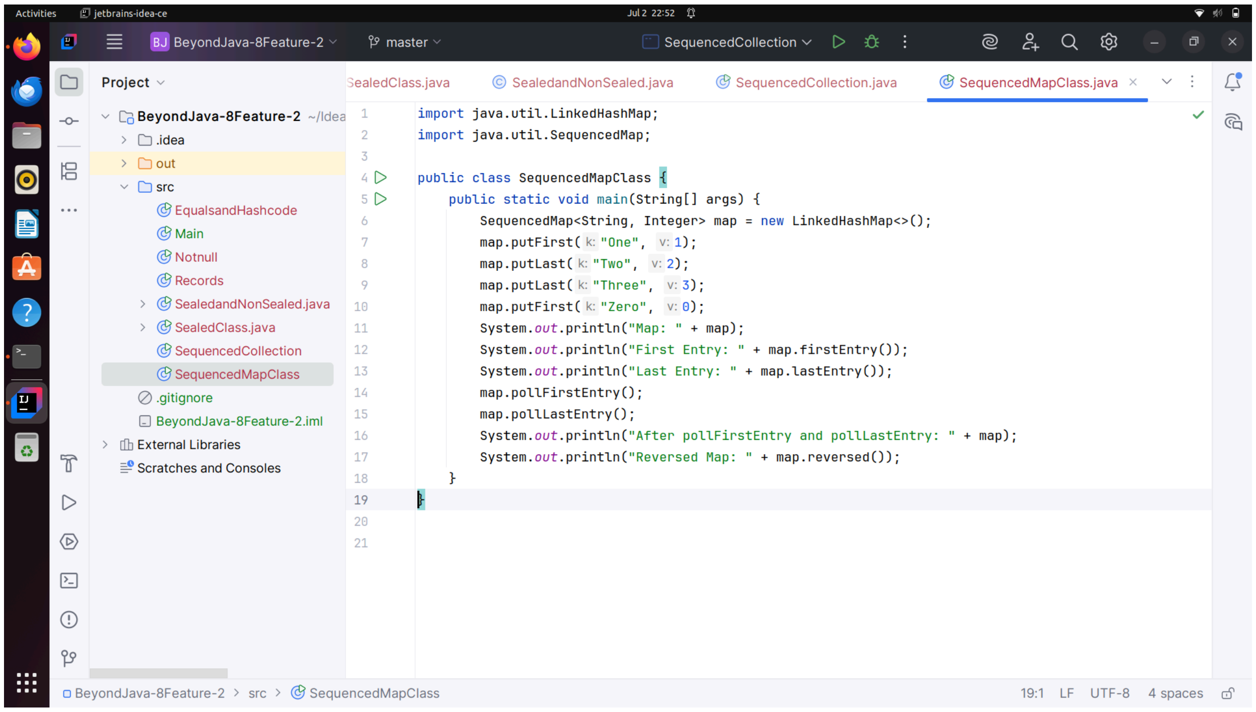
```java
import java.util.*;
public class SequencedCollection {
    public static void main(String[] args) {
        List<String> fruits = new LinkedList<>();
        fruits.addFirst("Banana");
        fruits.addLast("Apple");
        fruits.addFirst("Cherry");
        System.out.println("List: " + fruits);
        System.out.println("First: " + fruits.getFirst());
        System.out.println("Last: " + fruits.getLast());
        fruits.removeFirst();
        fruits.removeLast();
        System.out.println("After removeFirst & removeLast: " + fruits);
        System.out.println("Reversed List: " + fruits.reversed());
        Set<String> set = new LinkedHashSet<>();
        set.add("Red");
        set.add("Green");
        set.add("Blue");
        ((SequencedSet<String>) set).addFirst("Black");
        ((SequencedSet<String>) set).addLast("White");
        System.out.println("\nSet: " + set);
        System.out.println("First: " + ((SequencedSet<String>) set).getFirst());
        System.out.println("Last: " + ((SequencedSet<String>) set).getLast());
        ((SequencedSet<String>) set).removeFirst();
        ((SequencedSet<String>) set).removeLast();
        System.out.println("After removeFirst & removeLast: " + set);
        System.out.println("Reversed Set: " + ((SequencedSet<String>) set).reversed());
```

Output:



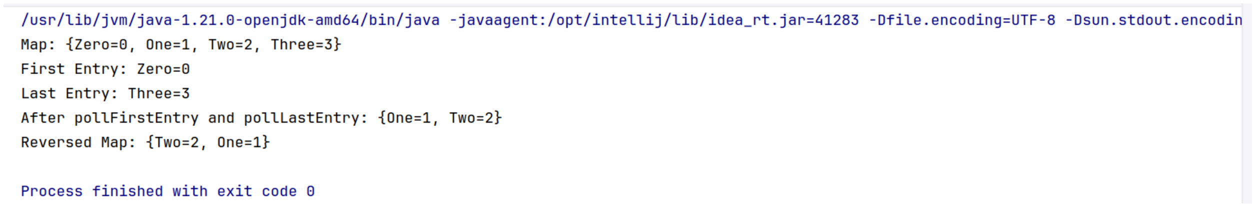
```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=45215 -Dfile.encoding=UTF-8 -Dsun.stdout.encodin
List: [Cherry, Banana, Apple]
First: Cherry
Last: Apple
After removeFirst & removeLast: [Banana]
Reversed List: [Banana]

Set: [Black, Red, Green, Blue, White]
First: Black
Last: White
After removeFirst & removeLast: [Red, Green, Blue]
Reversed Set: [Blue, Green, Red]

Process finished with exit code 0
```

7. Demonstrate the use of firstEntry(), lastEntry(), pollFirstEntry(), pollLastEntry(), putFirst(), putLast(), reversed() with SequencedMap.



```java
import java.util.LinkedHashMap;
import java.util.SequencedMap;

public class SequencedMapClass {
    public static void main(String[] args) {
        SequencedMap<String, Integer> map = new LinkedHashMap<>();
        map.putFirst( k: "One",  v: 1);
        map.putLast( k: "Two",  v: 2);
        map.putLast( k: "Three",  v: 3);
        map.putFirst( k: "Zero",  v: 0);
        System.out.println("Map: " + map);
        System.out.println("First Entry: " + map.firstEntry());
        System.out.println("Last Entry: " + map.lastEntry());
        map.pollFirstEntry();
        map.pollLastEntry();
        System.out.println("After pollFirstEntry and pollLastEntry: " + map);
        System.out.println("Reversed Map: " + map.reversed());
    }
}
```

Output:



```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=41283 -Dfile.encoding=UTF-8 -Dsun.stdout.encodin
Map: {Zero=0, One=1, Two=2, Three=3}
First Entry: Zero=0
Last Entry: Three=3
After pollFirstEntry and pollLastEntry: {One=1, Two=2}
Reversed Map: {Two=2, One=1}

Process finished with exit code 0
```