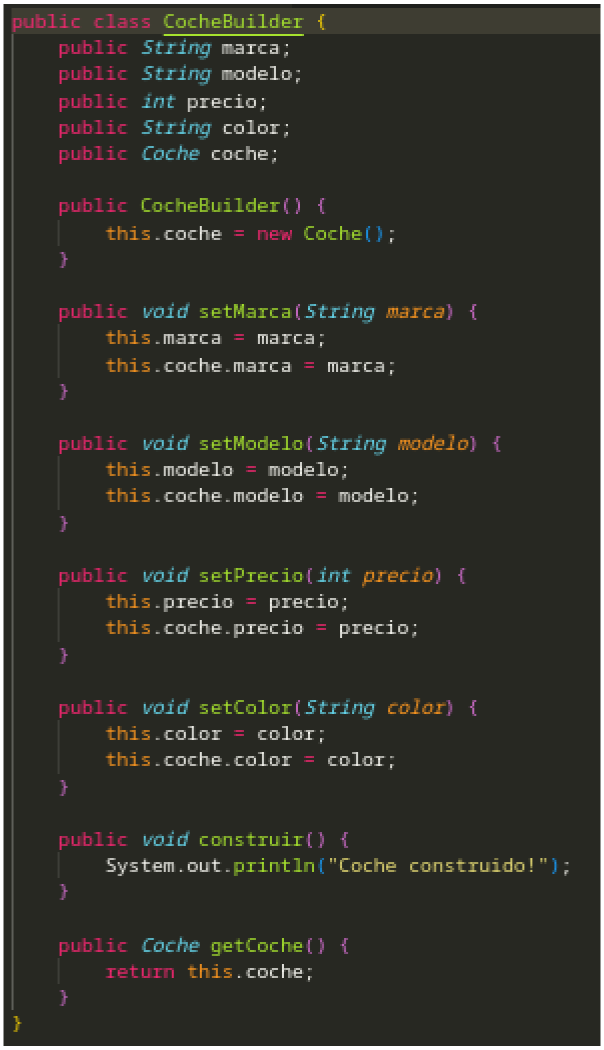
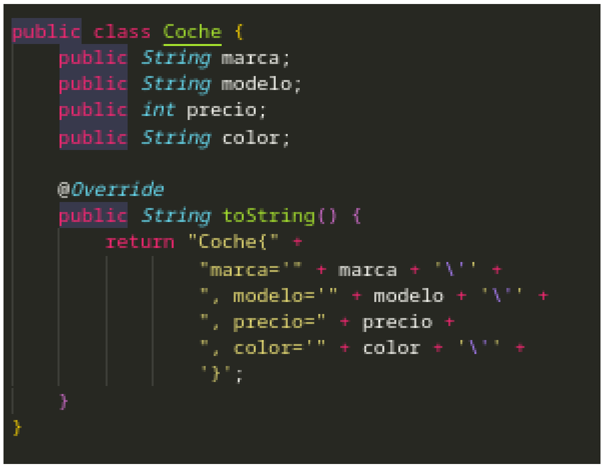
En cada uno de los siguientes programas, identifique si se usa el patrón builder o no y justifique el porqué.

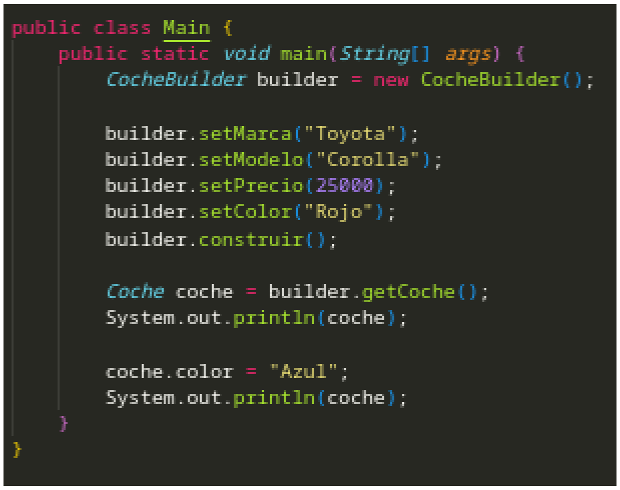
1. **Producto Coche**



```java
public class CocheBuilder {
    public String marca;
    public String modelo;
    public int precio;
    public String color;
    public Coche coche;

    public CocheBuilder() {
        this.coche = new Coche();
    }

    public void setMarca(String marca) {
        this.marca = marca;
        this.coche.marca = marca;
    }

    public void setModelo(String modelo) {
        this.modelo = modelo;
        this.coche.modelo = modelo;
    }

    public void setPrecio(int precio) {
        this.precio = precio;
        this.coche.precio = precio;
    }

    public void setColor(String color) {
        this.color = color;
        this.coche.color = color;
    }

    public void construir() {
        System.out.println("Coche construido!");
    }

    public Coche getCoche() {
        return this.coche;
    }
}
```

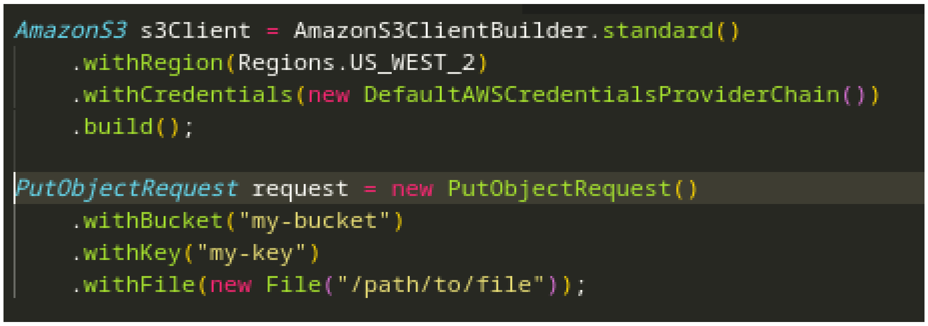


```java
public class Coche {
    public String marca;
    public String modelo;
    public int precio;
    public String color;

    @Override
    public String toString() {
        return "Coche{" +
                "marca='" + marca + '\'' +
                ", modelo='" + modelo + '\'' +
                ", precio=" + precio +
                ", color='" + color + '\'' +
                '}';
    }
}
```



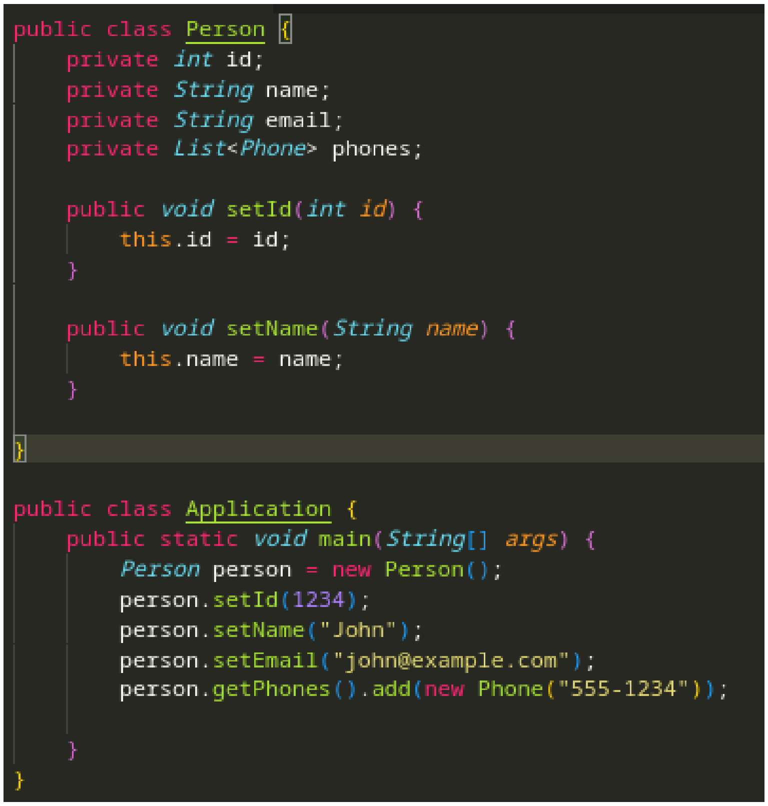
```java
public class Main {
    public static void main(String[] args) {
        CocheBuilder builder = new CocheBuilder();

        builder.setMarca("Toyota");
        builder.setModelo("Corolla");
        builder.setPrecio(25000);
        builder.setColor("Rojo");
        builder.construir();

        Coche coche = builder.getCoche();
        System.out.println(coche);

        coche.color = "Azul";
        System.out.println(coche);
    }
}
```

2. AWS



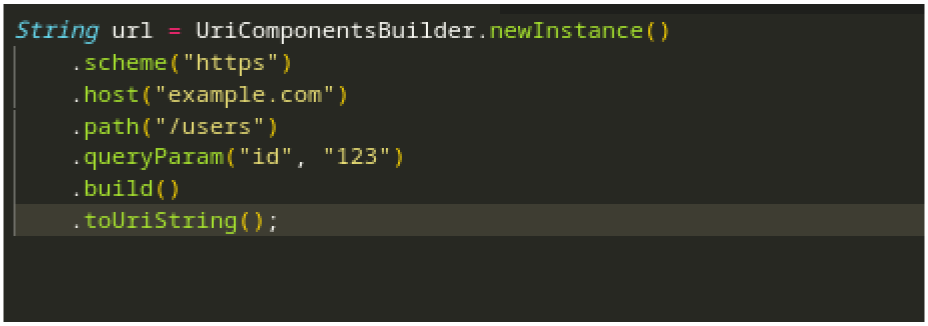
```java
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .withCredentials(new DefaultAWSCredentialsProviderChain())
    .build();

PutObjectRequest request = new PutObjectRequest()
    .withBucket("my-bucket")
    .withKey("my-key")
    .withFile(new File("/path/to/file"));
```

3. Person



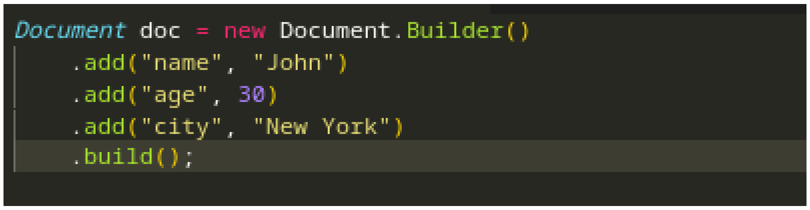
```java
public class Person {
    private int id;
    private String name;
    private String email;
    private List<Phone> phones;

    public void setId(int id) {
        this.id = id;
    }

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

}

public class Application {
    public static void main(String[] args) {
        Person person = new Person();
        person.setId(1234);
        person.setName("John");
        person.setEmail("john@example.com");
        person.getPhones().add(new Phone("555-1234"));

    }
}
```

4. SpringBoot Java



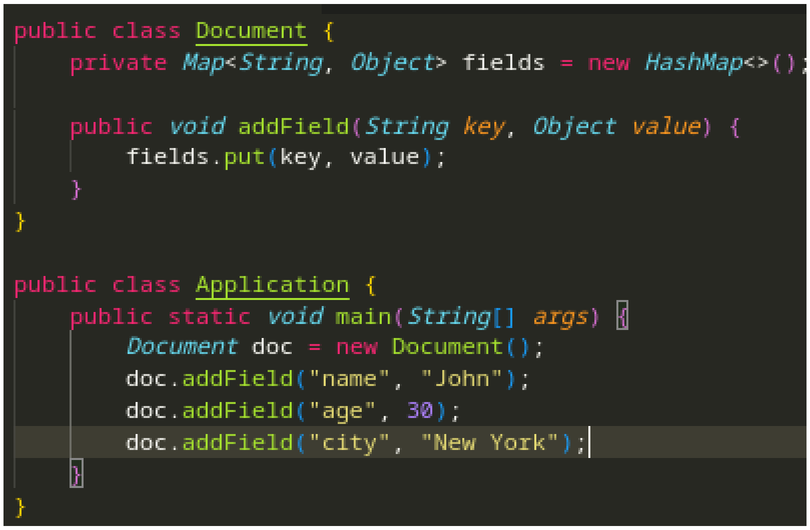
```java
String url = UriComponentsBuilder.newInstance()
    .scheme("https")
    .host("example.com")
    .path("/users")
    .queryParam("id", "123")
    .build()
    .toUriString();
```

5. MongoDB



```java
Document doc = new Document.Builder()
    .add("name", "John")
    .add("age", 30)
    .add("city", "New York")
    .build();
```

6. Document



```java
public class Document {
    private Map<String, Object> fields = new HashMap<>();

    public void addField(String key, Object value) {
        fields.put(key, value);
    }
}

public class Application {
    public static void main(String[] args) {
        Document doc = new Document();
        doc.addField("name", "John");
        doc.addField("age", 30);
        doc.addField("city", "New York");
    }
}
```