Here are 5 examples of testing Java objects using `assertEquals` in JUnit 5. These examples will demonstrate how to compare objects based on their fields or properties.

Example 1: Testing a Person Object

Step 1: Create the `Person` class

1. In the `src/main/java` folder, create a class `Person.java`.

```
public class Person {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return
false;
        Person person = (Person) obj;
        return age == person.age && name.equals(person.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, age);
    }
}
```

1.1. Person Class Explanation

The `Person` class has two attributes: `name` (String) and `age` (int). It also contains:

- A constructor to initialize these attributes.
- Getter methods for `name` and `age`.
- The `equals()` method to compare two `Person` objects based on their `name` and `age`.
- The `hashCode()` method to return a hash code based on `name` and `age`.

1.2. Main Class to Implement the Person Class Methods

```
import java.util.Objects;

public class Main {
    public static void main(String[] args) {
        // Step 1: Create two Person objects
        Person person1 = new Person("John Doe", 30);
        Person person2 = new Person("Jane Doe", 28);
        Person person3 = new Person("John Doe", 30);

        // Step 2: Print the details of the persons
        System.out.println("Person 1: " + person1.getName() + ", Age:
" + person1.getAge());
        System.out.println("Person 2: " + person2.getName() + ", Age:
" + person2.getAge());
        System.out.println("Person 3: " + person3.getName() + ", Age:
" + person3.getAge());

        // Step 3: Compare person1 with person2 (using equals method)
        if (person1.equals(person2)) {
            System.out.println("person1 and person2 are equal.");
        } else {
            System.out.println("person1 and person2 are not equal.");
        }

        // Step 4: Compare person1 with person3 (using equals method)
```

```
        if (person1.equals(person3)) {
            System.out.println("person1 and person3 are equal.");
        } else {
            System.out.println("person1 and person3 are not equal.");
        }

        // Step 5: Print hash codes of the persons
        System.out.println("Hash Code of person1: " +
person1.hashCode());
        System.out.println("Hash Code of person2: " +
person2.hashCode());
        System.out.println("Hash Code of person3: " +
person3.hashCode());
    }
}
```

1.3. Detailed Comments (Step-by-Step)

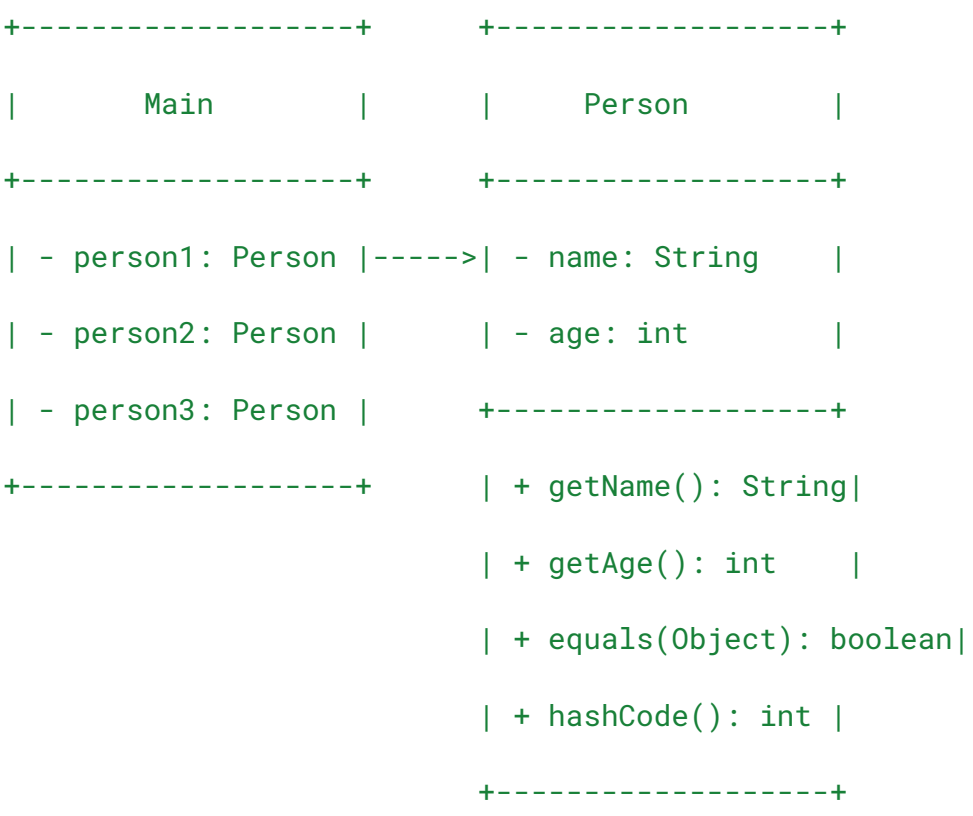
1. Step 1: Create two `Person` objects
    - In the `main` method, we create three `Person` objects: `person1`, `person2`, and `person3` using the `Person` constructor.
        - `person1` has the name "John Doe" and age 30.
        - `person2` has the name "Jane Doe" and age 28.
        - `person3` has the name "John Doe" and age 30 (same as `person1`).
        -
2. Step 2: Print the details of the persons
    - We call the `getName()` and `getAge()` methods on each `Person` object to display the name and age of each person.
    -
3. Step 3: Compare `person1` with `person2`
    - We use the `equals()` method to compare `person1` and `person2`. Since their names and ages are different, the method returns `false`, and the message "person1 and person2 are not equal." is printed.
    -
4. Step 4: Compare `person1` with `person3`

- We again use the `equals()` method to compare `person1` and `person3`. Since their names and ages are identical, the method returns `true`, and the message "person1 and person3 are equal." is printed.

5. Step 5: Print the hash codes
    - We use the `hashCode()` method to print the hash code values for `person1`, `person2`, and `person3`. This will be based on their `name` and `age` values.

1.4. Object Diagram

The object diagram shows how the `Person` objects are structured and related:



1.5. Algorithm (Step-by-Step)

1. Start the Program:
    - The program begins with the execution of the `main` method.
2. Create the `Person` Objects:
    - Three `Person` objects are created:
        - `person1` with name "John Doe" and age 30.
        - `person2` with name "Jane Doe" and age 28.
        - `person3` with name "John Doe" and age 30.
3. Print the Details of Each Person:
    - We call the `getName()` and `getAge()` methods to display the name and age of each `Person`.
4. Compare `person1` and `person2`:
    - Using the `equals()` method, we compare `person1` with `person2`. Since their names and ages are different, it returns `false`, and the message "person1 and person2 are not equal." is printed.
5. Compare `person1` and `person3`:
    - Using the `equals()` method again, we compare `person1` with `person3`. Since their names and ages are the same, it returns `true`, and the message "person1 and person3 are equal." is printed.
6. Print the Hash Codes:
    - The `hashCode()` method is called on each `Person` object, and the hash codes are printed. This will show the generated hash codes based on the `name` and `age` values.

1.6. Expected Output

1.

```
Person 1: John Doe, Age: 30

Person 2: Jane Doe, Age: 28
```

```
Person 3: John Doe, Age: 30

person1 and person2 are not equal.

person1 and person3 are equal.

Hash Code of person1: <calculated_hash>

Hash Code of person2: <calculated_hash>

Hash Code of person3: <calculated_hash>
```

Step 2: Create a JUnit test for the `Person` class

2. In the `src/test/java` folder, create a test class `PersonTest.java`.

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class PersonTest {

    @Test
    public void testPersonEquality() {
        Person person1 = new Person("Alice", 30);
        Person person2 = new Person("Alice", 30);
        assertEquals(person1, person2, "Persons with the same name
and age should be equal");
    }
}
```

Step 3: Run the test

1. Right-click on the `PersonTest.java` file and choose **Run 'PersonTest'**.
2. The test should pass, as the `Person` objects are equal based on the overridden `equals` method.

Example 2: Testing a Car Object

Step 1: Create the `Car` class

1. In the `src/main/java` folder, create a class `Car.java`.

```
public class Car {
    private String make;
    private String model;

    public Car(String make, String model) {
        this.make = make;
        this.model = model;
    }

    public String getMake() {
        return make;
    }

    public String getModel() {
        return model;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return
false;
        Car car = (Car) obj;
        return make.equals(car.make) && model.equals(car.model);
    }

    @Override
    public int hashCode() {
        return Objects.hash(make, model);
    }
```

```
}
```

Step 2: Create a JUnit test for the `Car` class

1. In the `src/test/java` folder, create a test class `CarTest.java`.

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class CarTest {

    @Test
    public void testCarEquality() {
        Car car1 = new Car("Toyota", "Camry");
        Car car2 = new Car("Toyota", "Camry");
        assertEquals(car1, car2, "Cars with the same make and model
should be equal");
    }
}
```

Step 3: Run the test

1. Right-click on the `CarTest.java` file and choose **Run 'CarTest'**.
2. The test should pass, as the `Car` objects are equal based on the overridden `equals` method.

Example 3: Testing a Book Object

Step 1: Create the Book class

1. In the `src/main/java` folder, create a class `Book.java`.

```
public class Book {
    private String title;
    private String author;

    public Book(String title, String author) {
        this.title = title;
        this.author = author;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return
false;
        Book book = (Book) obj;
        return title.equals(book.title) &&
author.equals(book.author);
    }

    @Override
    public int hashCode() {
        return Objects.hash(title, author);
    }
}
```

Step 2: Create a JUnit test for the `Book` class

1. In the `src/test/java` folder, create a test class `BookTest.java`.
2.

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class BookTest {

    @Test
    public void testBookEquality() {
        Book book1 = new Book("The Great Gatsby", "F. Scott
Fitzgerald");
        Book book2 = new Book("The Great Gatsby", "F. Scott
Fitzgerald");
        assertEquals(book1, book2, "Books with the same title and
author should be equal");
    }
}
```

Step 3: Run the test

1. Right-click on the `BookTest.java` file and choose **Run 'BookTest'**.
2. The test should pass, as the `Book` objects are equal based on the overridden `equals` method.

Example 4: Testing a User Object with Address

Step 1: Create the `Address` class

1. In the `src/main/java` folder, create a class `Address.java`.

```
public class Address {
    private String street;
    private String city;

    public Address(String street, String city) {
        this.street = street;
        this.city = city;
    }

    public String getStreet() {
        return street;
    }

    public String getCity() {
        return city;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return
false;
        Address address = (Address) obj;
        return street.equals(address.street) &&
city.equals(address.city);
    }

    @Override
    public int hashCode() {
        return Objects.hash(street, city);
    }
}
```

Step 2: Create the `User` class

1. In the `src/main/java` folder, create a class `User.java`.
2.

```
public class User {
    private String name;
    private Address address;

    public User(String name, Address address) {
        this.name = name;
        this.address = address;
    }

    public String getName() {
        return name;
    }

    public Address getAddress() {
        return address;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return
false;
        User user = (User) obj;
        return name.equals(user.name) &&
address.equals(user.address);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, address);
    }
}
```

Step 3: Create a JUnit test for the `User` and `Address` classes

1. In the `src/test/java` folder, create a test class `UserTest.java`.

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class UserTest {

    @Test
    public void testUserEquality() {
        Address address = new Address("123 Main St", "Springfield");
        User user1 = new User("John Doe", address);
        User user2 = new User("John Doe", address);
        assertEquals(user1, user2, "Users with the same name and
address should be equal");
    }
}
```

Step 4: Run the test

1. Right-click on the `UserTest.java` file and choose **Run 'UserTest'**.
2. The test should pass, as the `User` objects are equal based on the overridden `equals` method.

Example 5: Testing a Product Object

Step 1: Create the `Product` class

1. In the `src/main/java` folder, create a class `Product.java`.

```
public class Product {
    private String productName;
    private double price;

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

    public String getProductName() {
        return productName;
    }

    public double getPrice() {
        return price;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return
false;
        Product product = (Product) obj;
        return Double.compare(product.price, price) == 0 &&
productName.equals(product.productName);
    }

    @Override
    public int hashCode() {
        return Objects.hash(productName, price);
    }
}
```

Step 2: Create a JUnit test for the `Product` class

1. In the `src/test/java` folder, create a test class `ProductTest.java`.
2.

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class ProductTest {

    @Test
    public void testProductEquality() {
        Product product1 = new Product("Laptop", 1500.00);
        Product product2 = new Product("Laptop", 1500.00);
        assertEquals(product1, product2, "Products with the same name
and price should be equal");
    }
}
```

Step 3: Run the test

1. Right-click on the `ProductTest.java` file and choose **Run 'ProductTest'**.
2. The test should pass, as the `Product` objects are equal based on the overridden `equals` method.

Recap of Steps for Each Example:

1. **Create the class**: Define the class with necessary fields and methods, including the `equals` and `hashCode` methods.
2. **Write the test**: Write a test class with a test method annotated with `@Test`. Use `assertEquals(expected, actual)` to verify object equality.
3. **Run the test**: Right-click on the test class and choose **Run** to execute the test.

With these examples, you should be able to understand how to test objects using `assertEquals` in JUnit 5. The key is overriding the `equals` and `hashCode` methods to properly compare objects.