JavaBeans

Learning objectives

- JavaBean key concepts and conventions
- Overriding toString
- Overriding equals
- Overriding hashCode

JavaBeans

- Convention to create reusable, modular and customizable components
- JavaBeans follows a set of guidelines and conventions to create reusable and self-contained software components

JavaBeans - Key concepts

- Private instance variables to store state - helps encapsulation
- The private instance variables are by convention accessed by public getter methods and modified by public setter methods
- Default constructor - should provide a no-argument default constructor
- Serializable - often implement the Serializable interface allowing them to be serialized to be saved to a file or transmitted over a network
- No dependencies - JavaBeans should be self-contained and not have dependencies on other classes to promote reusability

Typical JavaBeans

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

    public Dog() {
    }

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

    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

Methods JavaBeans often override

- toString
- Equals
- hashCode
- These three methods are inherited from the Object class and often overridden for better functionality

Overriding toString

- toString is used to provide a String representation of an object
- The default behavior of the toString method inherited from Object is to print the class name, then @, then unsigned hexadecimal representation of the hash code of the object, for example: Dog@452b3a41
- Overriding toString gives us a chance to print something more meaningful as a representation of the state of the object

Demo 1 - Overriding toString

- The default implementation of toString
- Overriding toString to make it more useful

Exercise 1 - toString

- Create or reuse a class of type Book
- Create a book object and print it to the console
- Without overriding toString you should see something like this: Book@30f39991
- Override the toString method and you should see your own custom String representation of the instance values

Overriding equals

- The equals method should tell if two objects are equal or not when compared
- The default implementation of equals compare the object references of two variables, that is if they refer to the same object
- In many cases it would be more useful to define equality based on the state of the objects (the values of the instance variables)
- It’s a good programming practice to override equals and compare values in variables

Demo 2 - Overriding equals

- The default implementation of equals
- Overriding equals to make it more useful

Exercise 2 - equals

- Use the same Book class as in the previous exercise
- Create two identical books and compare if they are equal in the main method
- Without overriding the equals method the result should be that the books are not equal even though they have identical values in all the instance variables
- Override the equals method and compare the values of the instance variables and two identical books should now be equal

Overriding hashCode

- The hashCode method should generate a hash code based on the state of the object (the values in the instance variables)
- The hash code is a numeric identifier of an object
- Collections like HashMap and HashSet rely on the hash code to store and retrieve objects
- The equals and the hashCode of an object should be consistent, which means if two objects are equal they should also have the same hashCode
- If the equals method is overridden the hashCode method should also be overridden to ensure the consistency between the two methods

Demo 3 - Overriding hashCode

- The default implementation of hashCode
- Overriding hashCode to make it more useful and consistent with the equals method

Exercise 3 - hashCode

- Use the same Book class as in the previous exercise
- Create two identical books and print the hashCode of both books
- Without overriding the hashCode method the printouts should not be the same even if the books are equal according to the equals method
- Override the hashCode method and print the hashCodes of the two identical books, now the hashCodes should be the same

Learning objectives

- JavaBean key concepts and conventions
- Overriding toString
- Overriding equals
- Overriding hashCode