1) The Song class contains two equals() methods. Explain the difference using your knowledge about inheritance.

In Java, the equals() method is a method in the Object class, which is the superclass for all Java classes. Therefore, every class in Java inherits this method. The purpose of overriding the equals() method is to provide a way to compare objects of a class for equality based on their content rather than their reference (memory address).

In the Song class, there are two equals() methods:

- public boolean equals(Object o): This method overrides the equals() method from the Object class. Its purpose is to provide a way to compare a Song object with any other object. Inside this method, it first checks if the object provided is an instance of the Song class. If it is, it then delegates the comparison to the second equals() method (equals(Song s)). This is necessary because the equals(Object o) method must be able to accept any object type, per the contract of the Object class.

- public boolean equals(Song s): This is an overloaded version of the equals() method, specifically tailored for comparing two Song objects. It provides a more straightforward and type-safe way to compare Song objects, as it does not require type checking and casting which are needed in the equals(Object o) method. This method directly compares the relevant fields (title and instruments) of the two Song objects.

The existence of these two methods allows for both general-purpose equality checks (via equals(Object o)) and more specific, type-safe checks (via equals(Song s)).

2) Why do we need the version of equals() that takes an Object as the parameter in this class?

The version of the equals() method that takes an Object as a parameter is necessary for several reasons:

- Compatibility with Collections: Java collections such as ArrayList, HashSet, and others use the equals(Object o) method to compare elements. For instance, when checking if a collection contains an element, the collection's contains() method internally uses the equals(Object o) method. Without overriding this method in the Song class, these collections would use the default implementation from the Object class, which compares memory addresses and not the content of the objects.

- Fulfilling the Contract of the Object Class: By overriding equals(Object o), the Song class fulfills the contract of the Object class that dictates that this method must be able to handle comparisons with any object, not just Song objects. This is important for ensuring that the Song class can interact correctly with Java's framework and libraries, which rely on this method.

- Polymorphism: Overriding equals(Object o) ensures that the method behaves correctly even when Song objects are referred to by a superclass type reference, such as Object. This is crucial in polymorphic scenarios, where a Song object might be passed around in a context that only knows it as an Object.