Hyperiondev

TASK

Interview Preparation: Interfaces

Visit our website

Introduction

WELCOME TO THE INTERFACES INTERVIEW PREPARATION TASK!

The most important step in landing your dream job is acing your interview. Interviews allow you to impress your future employer by demonstrating your knowledge and skills. An interview can be a nerve-wracking process though. Taking the time to prepare for your interview can ensure that the process runs smoothly and is less stressful. In this task, we will be discussing interfaces.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact an expert code reviewer to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with a code reviewer. You can also schedule a call or get support via email.

Our expert code reviewers are happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

INTRODUCTION TO INTERFACES

By now you should be comfortable with the concept of interfaces. Below is a refresher of what they are and how they work for your second interview preparation task.

An interface is similar to a class, however, unlike classes, it can only contain method signatures and constants. A method signature is simply the name and the parameter types of a method, and not its implementation. An interface can be thought of as a blueprint of a class; they specify what a class must do but not how they go about doing it.

Interfaces are used to achieve full abstraction. Abstraction involves showing only the relevant data and hiding all unnecessary details of an object from the user. The methods in an interface do not contain bodies, they have to be implemented by a class before you can access them. The class that implements an interface must implement all of its methods.

Java uses the “interface” keyword to define an interface. The syntax used to define an interface is as follows:

```
public interface InterfaceName {
    //Constant declarations
    // Method signatures
}
```

Below is an example of an interface called Edible that that includes a single method named howToEat():

```
public interface Edible {
    // Describes how to eat
    public abstract String howToEat();
}
```

You can use this interface to specify whether an object is edible. This is accomplished by letting the class for the object implement this interface using the implements keyword. The name of an interface is an adjective (such as Edible) and not a noun. This is because interfaces specify some common behaviour for objects of the classes that implement the interface. Classes that implement the Edible interface represent objects that can be eaten.

Methods in an interface are assumed to be either public or abstract. In order to implement an interface, a class needs to specify an implements clause in its class declaration and provide an implementation for every method in the interface. The following is an example of a class that implements the Edible interface:

```
public class Apple implements Edible {

    public String howToEat() {
        return "Apple: Make apple juice";
    }
}
```

In the code above, the declaration for the Apple class specified implements Edible and the body of the class includes an implementation of the `howToEat()` method. When a class implements an interface, it implements all the methods defined in the interface with the exact signature and return type. A class can also implement more than one interface.

Just like a class, an interface is a data type and you can, therefore, use an interface as the type for a variable, parameter, or method return value. For example:

```
Edible food = getFood();
food.howToEat();
```

In the code above, the getFood() method returns an object that implements the Edible interface. The object is assigned to a variable called `food` of type `Edible`. In the second statement, the objects howToEat() method is called. You can also call the constructor of the class that implements the interface. For example:

```
Edible food = new Apple();
```

This was a brief introduction to interfaces. Using official **Oracle interfaces tutorial** or any other references you might find helpful, answer the questions in the compulsory task below.

BEFORE YOU START THE COMPULSORY TASK

We advise that you create a blog to answer the questions in the compulsory task. The reason for this is twofold:

- Blogs are usually “conversational” in the sense that the blogger usually writes in such a way that they are explaining a topic to the reader. Since these interview preparation tasks are meant to help you explain your knowledge of a topic to a prospective employer, a blog is a good format to help you get used to explaining technical concepts to others.
- Your blog post could, in itself, be a useful tool that you could use to impress your potential employer. Imagine this: your employer asks you about concurrency, maybe even something you don’t completely understand about the way a specific tool/language supports concurrency, but you can say, “It’s interesting that you ask that. I haven’t specifically worked with … but I actually have written a blog that explores concurrency with JavaScript and Python…” Impressive!!

If you don’t yet have a blog, you can learn how to create one **here**. If you would like to make your blog private at this stage, please make sure that at least your mentor has access to your blog by following these instructions **here**.

Compulsory Task

Answer the following questions:

- What is an interface in Java and why would you, as a software developer, use interfaces?
- What is the difference between an abstract class and an interface?
- Why is abstraction an important concept in software development and what role do interfaces play in abstraction?
- What must a class do in order to implement an interface?
- What is an abstract method?
- Can you instantiate an interface?
- Can you declare a constructor inside an interface? If not, why?
- Can we override an interface method with visibility that is not public?

Completed the task(s)?

Ask an expert to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.