

1)How to work with null pointer exception using Optional Class.
 2)Search and create three examples on Optional Class
 3)Write a program to store Product(pid,pname,price) class into
 arrayList and
 using Streams convert Product object into Map<pname,price>



1. Handling Null Pointer Exceptions Using the `Optional` Class

The `Optional` class in Java is a container object that may or may not contain a non-null value. It was introduced in Java 8 to handle `NullPointerException` more gracefully. Instead of returning `null`, methods can return an `Optional` object, which can then be checked if a value is present or not. This way, you can avoid `NullPointerException` and make your code more readable and less error-prone.

Here's a quick guide on how to use the `Optional` class:

- **Creating an `Optional` Object:**
 - `Optional.of(value)`: Creates an `Optional` if the value is non-null.
 - `Optional.empty()`: Returns an empty `Optional` object.
 - `Optional.ofNullable(value)`: Creates an `Optional` if the value is non-null; otherwise, it returns an empty `Optional`.
- **Checking for Values:**
 - `isPresent()`: Returns `true` if there is a value present; otherwise, `false`.
 - `ifPresent(Consumer)`: Executes the specified Consumer if a value is present.
- **Getting Values:**
 - `get()`: Returns the value if present; otherwise, throws `NoSuchElementException`.
 - `orElse(T other)`: Returns the value if present; otherwise, returns `other`.
 - `orElseGet(Supplier<? extends T> other)`: Returns the value if present; otherwise, returns the result of the `Supplier`.
 - `orElseThrow(Supplier<? extends X> exceptionSupplier)`: Returns the value if present; otherwise, throws an exception provided by `exceptionSupplier`.

2. Examples of Using the `Optional` Class

Example 1: Using `Optional.ofNullable()` and `orElse()`

```
java
import java.util.Optional;

public class OptionalExample1 {
    public static void main(String[] args) {
        String value = null;

        // Creating Optional using ofNullable
        Optional<String> optionalValue = Optional.ofNullable(value);

        // Using orElse to provide a default value
        String result = optionalValue.orElse("Default Value");
        System.out.println(result); // Output: Default Value
    }
}
```

Example 2: Using `ifPresent()`

```
java
import java.util.Optional;

public class OptionalExample2 {
    public static void main(String[] args) {
        Optional<String> optionalValue = Optional.of("Hello, World!");

        // Using ifPresent to execute code if value is present
        optionalValue.ifPresent(value -> System.out.println("Value is: " + value));
        // Output: Value is: Hello, World!
    }
}
```

Example 3: Using `orElseThrow()`

```
java
import java.util.Optional;
```

```

import java.util.Optional;

public class OptionalExample3 {
    public static void main(String[] args) {
        Optional<String> optionalValue = Optional.empty();

        // Using orElseThrow to throw an exception if no value is present
        try {
            String result = optionalValue.orElseThrow(() -> new IllegalArgumentException("No value present"));
        } catch (Exception e) {
            System.out.println(e.getMessage()); // Output: No value present
        }
    }
}

```

3. Program to Store `Product` Class Objects in an `ArrayList` and Convert to `Map<pname, price>`

Let's create a `Product` class and a program to demonstrate storing objects in an `ArrayList` and converting them to a `Map` using Java Streams.

```

java Copy code

import java.util.*;
import java.util.stream.Collectors;

class Product {
    private int pid;
    private String pname;
    private double price;

    public Product(int pid, String pname, double price) {
        this.pid = pid;
        this.pname = pname;
        this.price = price;
    }

    public int getPid() {
        return pid;
    }

    public String getPname() {
        return pname;
    }

    public double getPrice() {
        return price;
    }

    @Override
    public String toString() {
        return "Product{" +
            "pid=" + pid +
            ", pname='" + pname + '\'' +
            ", price=" + price +
            '}';
    }
}

public class ProductExample {
    public static void main(String[] args) {
        // Creating an ArrayList of Product objects
        List<Product> products = new ArrayList<>();
        products.add(new Product(1, "Laptop", 1200.0));
        products.add(new Product(2, "Smartphone", 800.0));
        products.add(new Product(3, "Tablet", 400.0));

        // Converting List of Products to Map<pname, price>
        Map<String, Double> productPriceMap = products.stream()
            .collect(Collectors.toMap(Product::getPname, Product::getPrice));

        // Printing the Map
        productPriceMap.forEach((pname, price) ->
            System.out.println("Product Name: " + pname + ", Price: " + price));
    }
}

```

Explanation


- **Product Class:** Defines a product with `pid`, `pname`, and `price`.
- **ArrayList of Products:** We create an `ArrayList` to store `Product` objects.
- **Streams and Collectors:** Using Java Streams, we convert the list of products into a `Map` where

- **Streams and Collectors:** Using `zoo.streams`, we convert the list of products into a `map`, where the key is the product name (`"pname"`), and the value is the price (`"price"`).
- **Output:** The `forEach` method prints out each entry in the map.

Output

```
yaml Copy code  
  
Product Name: Laptop, Price: 1200.0  
Product Name: Smartphone, Price: 800.0  
Product Name: Tablet, Price: 400.0
```

 Message ChatGPT

ChatGPT can make mistakes. Check important info.

