**Solution for Assignment on Classes and Objects**

1. Person class:

import lombok.\*;

@Getter

@Setter

@AllArgsConstructor

@NoArgsConstructor

@ToString

public class Person {

@NonNull

private String name;

private int age;

private String occupation;

}

1. The **@Field** annotation is not a valid annotation for generating getters and setters. Instead, we can use Lombok annotations like **@Getter** and **@Setter**, which will automatically generate the getters and setters for us. These annotations are already included in the **Person** class above.

@Override

public String toString() {

return "Person(name=" + name + ", age=" + age + ", occupation=" + occupation + ")";

}

1. **toString()** method for Person class:

@Override

public String toString() {

return "Person(name=" + name + ", age=" + age + ", occupation=" + occupation + ")";

}

1. PersonService class:

public class PersonService {

public Person createPerson(String name, int age, String occupation) {

return new Person(name, age, occupation);

}

public void printPerson(Person person) {

System.out.println(person.toString());

}

}

1. **createPerson()** method in **PersonService** class:

public Person createPerson(String name, int age, String occupation) {

if (name.isEmpty()) {

throw new IllegalArgumentException("Name cannot be empty.");

}

if (age <= 0) {

throw new IllegalArgumentException("Age must be greater than zero.");

}

return new Person(name, age, occupation);

}

1. **printPerson()** method in **PersonService** class:

public void printPerson(Person person) {

System.out.println(person.toString());

}

1. Creating a **Person** object in the **Main** class:

public class Main {

public static void main(String[] args) {

PersonService personService = new PersonService();

Person johnDoe = personService.createPerson("John Doe", 30, "Software Developer");

personService.printPerson(johnDoe);

}

}

1. Output:

Person(name=John Doe, age=30, occupation=Software Developer)

1. Creating a new **Person** object with different properties and printing its string representation:

public class Main {

public static void main(String[] args) {

PersonService personService = new PersonService();

Person janeDoe = personService.createPerson("Jane Doe", 25, "Doctor");

personService.printPerson(janeDoe);

}

}

Output:

Person(name=Jane Doe, age=25, occupation=Doctor)

1. Updated **createPerson()** method in **PersonService** class:

public Person createPerson(String name, int age, String occupation) {

if (name.isEmpty()) {

throw new IllegalArgumentException("Name cannot be empty.");

}

if (age <= 0) {

throw new IllegalArgumentException("Age must be greater than zero.");

}

return new Person(name, age, occupation);

}