|  |  |
| --- | --- |
| A logo of a building  Description automatically generated | Information Technology Department - State Polytechnic of Malang  **Jobsheet-7: Overloading and Overriding**  **Course: Object Oriented Programming Practicum**  Lecturer: Endah Septa Sintiya, S.Pd., M.Kom.  *October 2024* |

Name: Azkiya Putri Ihtiarilah

Nim: 2341760175

Github: <https://github.com/azkiyaihtiarilah>

1. **Competence**

After taking this subject, students are able to:

* 1. Understand the concepts of overloading and overriding,
  2. Understand the difference between overloading and overriding,
  3. Accuracy in identifying overriding and overloading methods
  4. Accuracy in practicing instructions on the jobsheet
  5. Implement overloading and overriding methods.

1. **Introduction**
   1. **Overloading**

is to rewrite a method with the same name on a class. The goal is to facilitate the use/invocation of methods with similar functionality. The Overloading method declaration rules are as follows:

* + - The method name must be the same.
    - The list of parameters should be different.
    - The return type can be the same, or it can be different.

There are several lists of parameters on overloading can be seen as follows:

* + - The difference in the list of parameters does not only occur in the difference in the number of parameters, but also in the order of the parameters.
    - For example, the following two parameters:
      * Function\_member (int x, string n)
      * Function\_member (String n, int x)
    - The two parameters are also considered different in the list of parameters.
    - The parameter list is not related to the naming of the variables present in the parameter.
    - For example, the following 2 list of parameters:
      * function\_member(int x)
      * function\_member(int y)
    - The two lists of parameters above are considered the same because the only difference is the naming of the variable parameters.

Overloading can also occur between the parent class and its subclass if it meets all three overload conditions. There are several overloading rules, namely:

* + - Primitive widening conversions take precedence over overloading over boxing and var args.
    - We can't do the widening process from one wrapper type to another (changing the Integer to Long).
    - We can't do the widening process followed by boxing (from int to Long)
    - We can do boxing followed by widening (int can be an Object via an Integer)
    - We can combine var args with either widening or boxing
  1. **Overriding**

is a Subclass that seeks to modify behaviors inherited from super classes. The goal is that the subclass can have more specific behavior so that it can be done by redeclaring the parent class's method in the subclass.

The method declaration in the subclass must be the same as the one in the super class. Similarities on:

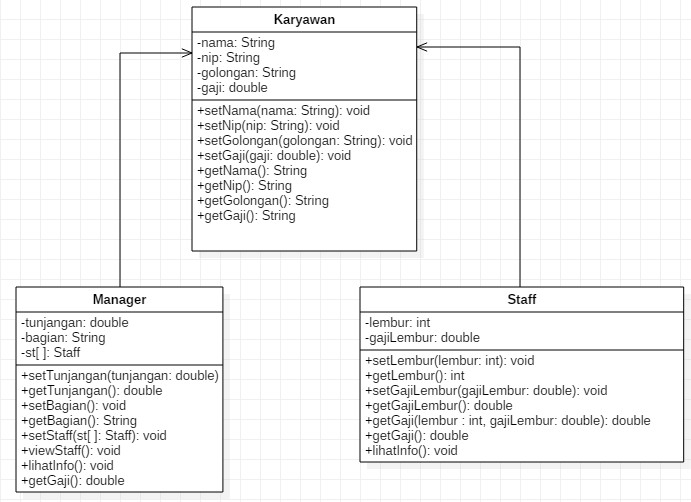
* + - Name
    - Return type (for return type: class A or is a subclass of class A)
    - List of parameters (number, type and order)

So that the method in the parent class is called the overridden method and the method in the subclass is called the overriding method. There are several method rules in overriding:

* + - The access mode of the overriding method must be the same or broader than the overridden method.
    - A subclass can only override a superclass method once, there must not be more than one method in the exact same class.
    - The overriding method must not throw checked exceptions that are not declared by the overridden method.

1. **Practicum**
   1. **Experiment 1**

For the following example case, there are three classes, namely Karyawan, Manager, and Staff. Employee Class is a superclass of Manager and Staff where the Manager and Staff subclasses have different methods for calculating salaries.



* 1. **Karyawan**

A screen shot of a computer

Description automatically generated

* 1. **Staff**

A screen shot of a computer program

Description automatically generated

* 1. **Manager**

A screen shot of a computer screen

Description automatically generated

* 1. **Main**

A screen shot of a computer

Description automatically generated

* 1. **Output**

**A screenshot of a computer program

Description automatically generated** **A screenshot of a computer

Description automatically generated**

1. **Exercise**

A screen shot of a computer program

Description automatically generated

**A screen shot of a computer

Description automatically generated**

* 1. From the source coding above, where is the overloading?

Overloading pada kode di atas terjadi pada metode perkalian. Overloading adalah konsep dalam pemrograman di mana sebuah metode memiliki nama yang sama tetapi parameter yang berbeda, baik dari segi jumlah atau tipe data.

* 1. If there is overloading, how many different parameters are there?

Terdapat dua parameter yang berbeda, yaitu:

|  |
| --- |
| Method pertama: |
| void perkalian(int a, int b){          System.out.println(a \* b);      } |
| Method kedua: |
| void perkalian(int a, int b, int c){          System.out.println(a \* b \* c);      } |

A screen shot of a computer program

Description automatically generated **A screen shot of a computer screen

Description automatically generated**

* 1. From the source coding above, where is the overloading?

Overloading pada kode di atas terjadi pada metode perkalian.

* 1. If there is overloading, how many different types of parameters are there?

|  |
| --- |
| Method pertama: |
| void perkalian(int a, int b){          System.out.println(a \* b);      } |
| Method kedua: |
| void perkalian(double a, double b){          System.out.println(a \* b);      } |

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A computer screen with white text

Description automatically generated

* 1. From the source coding above, where is the overriding?

Overriding adalah proses di mana subclass menyediakan implementasi khusus dari metode yang sudah didefinisikan di superclass-nya. overriding terjadi di kelas Piranha, di mana metode swim() dari kelas Ikan di-override.

* 1. Describe when sourcoding above if there is overriding?

|  |
| --- |
| Superclass (Ikan) memiliki metode swim |
| public void swim() {          System.out.println("Ikan bisa berenang");      } |
| Subclass (Piranha) meng-override metode swim |
| public void swim(){          System.out.println("Piranha bisa makan daging");      } |

|  |
| --- |
| Main class Fish |
| public static void main(String[] args) {          Ikan b = new Piranha();          b.swim();      } |

Ketika objek b dideklarasikan sebagai tipe Ikan tetapi diinstansiasi sebagai Piranha. Pemanggilan b.swim() akan menjalankan metode swim() milik kelas Piranha, bukan milik kelas Ikan, karena metode swim() telah di-override di subclass Piranha.

1. **Tasks**
   1. **Overloading**

Implement the overloading concept in the diagram class below:

A screenshot of a math equation

Description automatically generated with medium confidence

A screen shot of a computer program

Description automatically generated

Hasil Running Program:

A screen shot of a computer

Description automatically generated

* 1. **Overriding**

Implement the diagram class below using the dynamic method dispatch technique:

A diagram of a computer

Description automatically generated

Hasil Running Program:

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated