CONCEPT OF OBJECT ORIENTED PROGRAMMING

* 1. **Polymorphism**
     1. Define Polymorphism

Polimorfisme adalah keupayaan suatu kaedah untuk melakukan perkara-perkara yang berbeza berdasarkan objek yang ia bertindak. Dengan kata lain, polimorfisme membolehkan anda menentukan satu antara muka dan mempunyai pelbagai pelaksanaan. Seperti yang telah kita lihat dalam contoh di bawah bahawa kita telah menentukan bunyi kaedah () dan mempunyai banyak pelaksanaannya dalam kelas sub-2 yang berbeza.

Kaedah bunyi () yang akan dipanggil ditentukan semasa runtime supaya contoh yang kami berikan di bawah adalah contoh polimorfisme runtime.

*public class Animal{*

*...*

*public void sound(){*

*System.out.println("Animal is making a sound");*

*}*

*}*

Sekarang mari kita katakan kami dua subkelas kelas Haiwan: Kuda dan Kucing yang memanjangkan (lihat Warisan) kelas Haiwan. Kami boleh menyediakan pelaksanaan dengan kaedah yang sama seperti ini:

**Contoh :**

*public class Horse extends Animal{*

*...*

*@Override*

*public void sound(){*

*System.out.println("Neigh");*

*}*

*}*

**and**

*public class Cat extends Animal{*

*...*

*@Override*

*public void sound(){*

*System.out.println("Meow");*

*}*

*}*

* + 1. List the advantages of polymorphism
* Ia membantu pengaturcara menggunakan semula kod dan kelas sekali ditulis, diuji dan dilaksanakan. Ia boleh digunakan semula dalam pelbagai cara.
* Nama pemboleh ubah tunggal boleh digunakan untuk menyimpan pembolehubah pelbagai jenis data (Float, double, Long, Int dll).
* Polimorfisme membantu dalam mengurangkan gandingan antara fungsi yang berbeza.
  + 1. Apply Method and Constructur Overloading

**Method** adalah ciri yang membolehkan kelas mempunyai lebih daripada satu kaedah yang mempunyai nama yang sama, jika senarai hujah mereka berbeza. Ia sama dengan pembebanan overloading di Java, yang membolehkan kelas mempunyai lebih daripada satu pembina yang mempunyai senarai hujah yang berbeza.

mari kita kembali ke titik, apabila saya mengatakan senarai argumen itu bermakna parameter yang mempunyai kaedah: Sebagai contoh, senarai argumen kaedah tambah (int a, int b) yang mempunyai dua parameter adalah berbeza daripada senarai argumen kaedah tambah (int a, int b, int c) mempunyai tiga parameter.

Untuk overloading kaedah ini, senarai argumen kaedah mesti berbeza sama ada antara berikut:

1. **Number of parameters.**

Sebagai contoh: Ini adalah kes yang berlebihan

add (int, int)

add (int, int, int)

2. **Data type of parameters.**

For example:

add(int, int)

add(int, float)

3. **Sequence of Data type of parameters.**

For example:

add(int, float)

add(float, int)

**Contoh**

*class DisplayOverloading*

*{*

*public void disp(char c)*

*{*

*System.out.println(c);*

*}*

*public void disp(char c, int num)*

*{*

*System.out.println(c + " "+num);*

*}*

*}*

*class Sample*

*{*

*public static void main(String args[])*

*{*

*DisplayOverloading obj = new DisplayOverloading();*

*obj.disp('a');*

*obj.disp('a',10);*

*}*

*}*

**Output:**

a

a 10

Dalam contoh di atas - kaedah disp () terlalu banyak berdasarkan bilangan parameter - Kami mempunyai dua kaedah dengan nama disp tetapi parameter yang mereka ada berbeza. Kedua-duanya mempunyai bilangan parameter yang berbeza

**Constructur Overloading** adalah konsep yang mempunyai lebih daripada satu pembina dengan senarai parameter yang berbeza, sedemikian rupa supaya setiap pembina melakukan tugas yang berbeza. Contohnya. Kelas vektor mempunyai 4 jenis konstruktor. Jika anda tidak mahu menentukan kapasiti permulaan dan kenaikan kapasiti maka anda hanya boleh menggunakan pembina default kelas Vektor seperti Vector v = Vector baru (); Walau bagaimanapun, jika anda perlu menentukan kapasiti dan kenaikan maka anda panggil pembolehubah parameter kelas Vektor dengan dua argumen int seperti ini: Vektor v = Vektor baru (10, 5);

**Let’s understand the role of this () in constructor overloading**

*public class OverloadingExample2*

*{*

*private int rollNum;*

*OverloadingExample2()*

*{*

*rollNum =100;*

*}*

*OverloadingExample2(int rnum)*

*{*

*this();*

*/\*this() is used for calling the default*

*\* constructor from parameterized constructor.*

*\* It should always be the first statement*

*\* inside constructor body.*

*\*/*

*rollNum = rollNum+ rnum;*

*}*

*public int getRollNum() {*

*return rollNum;*

*}*

*public void setRollNum(int rollNum) {*

*this.rollNum = rollNum;*

*}*

*public static void main(String args[])*

*{*

*OverloadingExample2 obj = new OverloadingExample2(12);*

*System.out.println(obj.getRollNum());*

*}*

*}*

**Output**

*112*

**Sumber Rujukan**

<https://beginnersbook.com/2013/03/polymorphism-in-java/>