### MODUL 3

### *CONSTRUCTOR*

1. **Tujuan**
2. Mahasiswa memahami penggunaan *constructor* dalam OOP.
3. Mahasiswa memahami penggunaan *overloading*.
4. Mahasiswa memahami penggunaan *overloading* *constructor*.
5. **Dasar Teori**
6. ***Constructor***

Dalam bahasa pemrograman java *constructor* adalah blok kode yang mirip dengan *method*. *Constructor* digunakan untuk menginstansiai atau menciptakan sebuah objek. Nama *constructor* sama dengan nama kelas. *Constructor* tidak bisa mengembalikan nilai. Tanpa membuat *constructor* secara eksplisit-pun, Java akan menambahkan *constructor default* secara implisit. Tetapi jika kita sudah mendefinisikan minimal sebuah *constructor*, maka Java tidak akan menambah *constructor default*.

*Constructor default* tidak punya parameter. *Constructor* bisa digunakan untuk membangun suatu objek, langsung mengatur atribut-atributnya. *Constructor* seperti ini harus memiliki parameter masukkan untuk mengatur nilai atribut. *Access Modifier constructor* selayaknya adalah *public*, karena *constructor* akan diakses di luar kelasnya. Cara panggil *constructor* adalah dengan menambahkan *keyword “new”*. *Keyword new* dalam deklarasi ini artinya kita mengalokasikan pada memori sekian blok memori untuk menampung objek yang baru kita buat.

Karakteristik *constructor*:

* *Method constructor* harus memiliki nama yang sama dengan nama *class*
* Tidak mengembalikan suatu nilai (tidak ada *keyword return*)
* Dapat ditambah *access modifier public, private, protected* maupun *default*
* Suatu *constructor* bisa dipanggil oleh *constructor* lain dalam satu kelas.
  1. Penggunaan *Constructor*

Berikut adalah contoh penggunaan *constructor.*

* Buat *Class* *Bank*, isinya seperti berikut

|  |
| --- |
| public class Bank{  double saldo;  //konstruktor dengan parameter  public Bank(double Saldo){  this.saldo = Saldo;  }  void getSaldo(){  System.out.println("Saldo saat ini: "+saldo);  }  } |

* Pada *main class* nya kita isi seperti ini

|  |
| --- |
| public class JavaApplication{  public static void main(String[] args){  //bannku => objek  //Bank(100000) => konstruktornya  Bank bankku = new Bank(100000);  System.out.println("Saldo saat ini: "+bankku.saldo+ "\n");  Bankku.getSaldo();  }  } |

* 1. Konstruktor *Overriding & Overloading*
* *Overriding*

Subclass yang berusaha memodifikasi tingkah laku yang diwarisi dari *superclass*. Dilakukan dengan cara mendeklarasikan kembali *method* milik *parent class* di *subclass.*

Deklarasi *method* pada *subclass* harus sama dengan yang terdapat di *superclass*. Kesamaan pada :

* 1. Nama
  2. *Return type*
  3. Daftar parameter (jumlah, tipe, dan urutan)
  4. *Method* pada *parent class* disebut *overriden method*
  5. *Method* pada *subclass* disebut *overriding method*

Contoh *Overriding*

* *Class* *Animal* memiliki *method* SetVoice()

|  |
| --- |
| public class Animal {  public static SetVoice() {  System.out.println("Blesepblesep");  }  } |

* *Class* *Dog* yang merupakan turunan dari *Class* *Animal,* dan *class* ini memiliki *method* SetVoice() sama seperti *superclassnya* (*class* *Animal*). Ini disebut *overriding*

|  |
| --- |
| public class Dog extends Animal {  @Override  public void SetVoice(){  System.out.println("Hug hug");  }    } |

* *Overloading*

Menuliskan kembali *method* dengan nama yang sama pada suatu class. Ciri-ciri *overloading* :

* + - * 1. Nama *method* harus sama
        2. Daftar parameter harus berbeda
        3. *Return type* boleh sama, juga boleh berbeda

Tujuan: memudahkan penggunaan/pemanggilan *method* dengan fungsionalitas yang mirip.

Contoh *Overloading*

|  |
| --- |
| public class Bentuk {  public void Gambar(int t1) {  }  public void Gambar(int t1, int t2) {  }  public void Gambar(int t1, int t2, int t3) {  }  public void Gambar(int t1, int t2, int t3, int t4) {  }  } |

1. **Manfaat *Constructor***

*Constructor* biasa dipakai untuk membuat proses awal dalam persiapan object, seperti memberi nilai kepada *property*, memanggil *method internal* serta beberapa proses lain yang dirasa perlu. Dalam bahasa Java, *constructor* dibuat dengan cara menulis sebuah *method* yang namanya sama dengan nama *class.*

Sebuah *constructor* tidak mengembalikan nilai sehingga tidak perlu menulis tipe data sebelum nama *function*. *Constructor* juga harus memiliki hak akses ***public***, yang apabila tidak ditulis dianggap *public* secara default.

1. **Perbedaan *Constructor* dengan *Method***

Seperti yang sudah disebutkan sebelumnya, *constructor* dan *method* dalam Java memiliki kemiripan. *Method* Java adalah potongan kode yang memiliki beberapa nama tertentu. *Method* dapat dipanggil selama titik mana pun dalam program hanya dengan menggunakan nama *method*. *Method* juga dapat dipahami sebagai *subprogram* yang beroperasi pada data dan mengembalikan beberapa nilai.

*Constructor* Java adalah jenis *method* khusus. Keduanya mirip dalam banyak hal, tetapi tidak identik. Berikut adalah beberapa perbedaan terpenting antara *constructor* dan *method*:

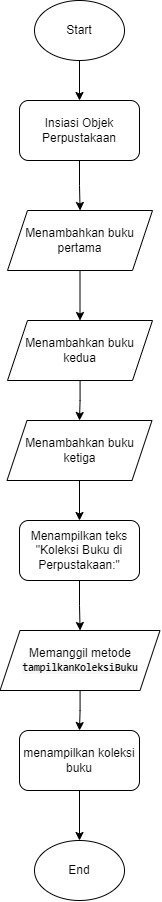
* *Invoke* – Saat *constructor* dipanggil secara implisit, *method* ini dipanggil secara eksplisit
* Java Kompilator – Kompilator Java tidak pernah menyediakan *method*. Namun, kompilator Java menyediakan *constructor default* jika tidak ditentukan dalam *class*
* *Number of Calls* – Nama *constructor* harus sama dengan nama *class*. Namun, *method* tersebut mungkin memiliki atau tidak memiliki nama yang sama dengan *class* yang memuatnya
* Jumlah Panggilan – *Constructor* dipanggil sekali dan hanya selama waktu pembuatan objek. *Method* di sisi lain, dapat dipanggil sebanyak yang diperlukan
* *Return Type* – *Method* harus memiliki jenis pengembalian, tetapi tidak wajib untuk *constructor*
* *Use* – Saat *method* digunakan untuk mengekspos perilaku objek, *constructor* digunakan untuk menginisialisasi keadaan yang sama.

1. **Studi Kasus / Permasalahan**
   * + 1. Buatlah aplikasi perpustakaan dengan *Constructor.*

**D. Hasil (*flowchart, script* program, hasil program)**

Buatlah aplikasi perpustakaan dengan *Constructor.*

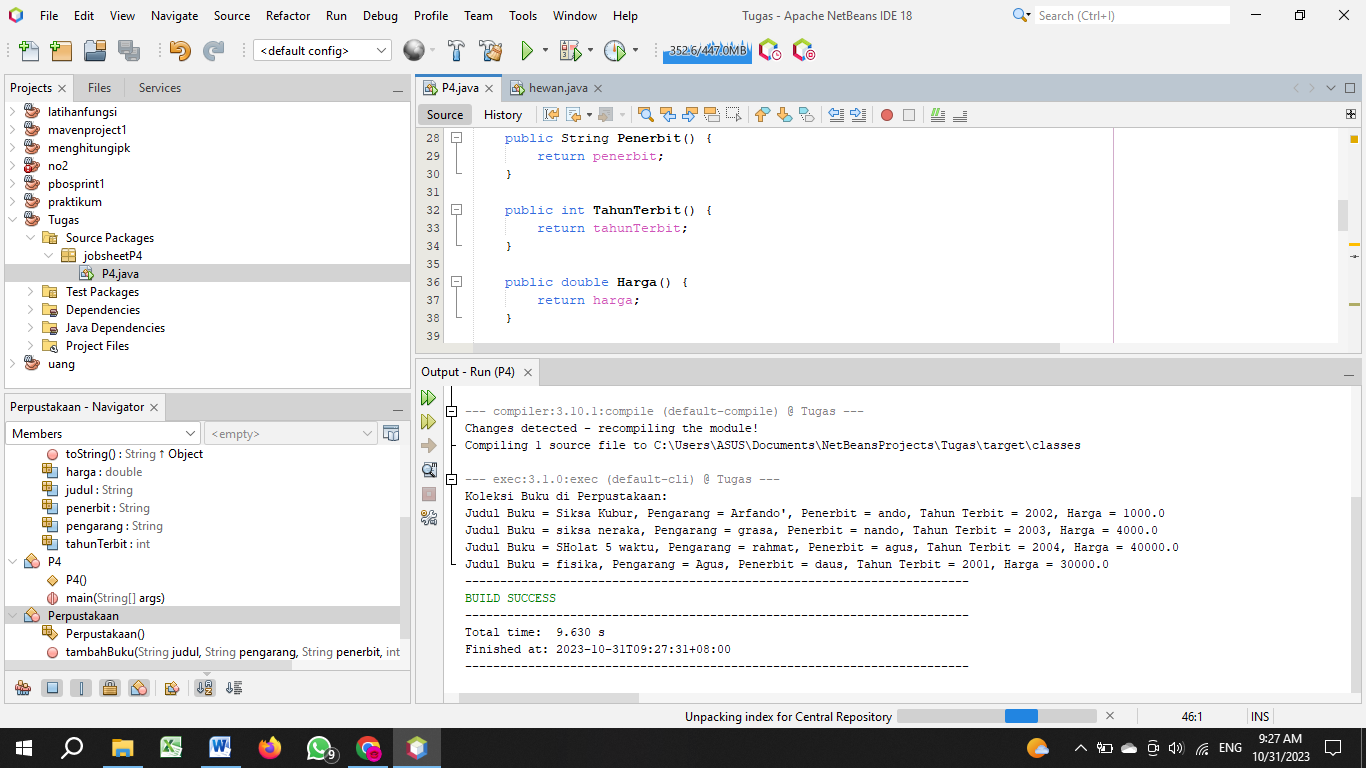
* Flowchart



* Script

|  |
| --- |
| package jobsheetP4;  import java.util.ArrayList;  class Buku {  String judul;  String pengarang;  String penerbit;  int tahunTerbit;  double harga;  public Buku(String j, String p, String pen, int tahun, double h) {  judul = j;  pengarang = p;  penerbit = pen;  tahunTerbit = tahun;  harga = h;  }  public String Judul() {  return judul;  }  public String Pengarang() {  return pengarang;  }  public String Penerbit() {  return penerbit;  }  public int TahunTerbit() {  return tahunTerbit;  }  public double Harga() {  return harga;  }  public String toString() {  return "Judul Buku = " + judul + ", Pengarang = " + pengarang + ", Penerbit = " + penerbit + ", Tahun Terbit = " + tahunTerbit + ", Harga = " + harga;  }  }  class Perpustakaan {  private final ArrayList<Buku> koleksiBuku = new ArrayList<>();  public void tambahBuku(String judul, String pengarang, String penerbit, int tahunTerbit, double harga) {  Buku agus = new Buku(judul, pengarang, penerbit, tahunTerbit, harga);  koleksiBuku.add(agus);  }  public void tampilBuku() {  for (Buku buku : koleksiBuku) {  System.out.println(buku);  }  }  }  public class P4 {  public static void main(String[] args) {  Perpustakaan agus = new Perpustakaan();  agus.tambahBuku("Siksa Kubur", "Arfando'", "ando", 2002, 1000);  agus.tambahBuku("siksa neraka", "grasa", "nando", 2003, 4000);  agus.tambahBuku("SHolat 5 waktu", "rahmat", "agus", 2004, 40000);  agus.tambahBuku("fisika", "Agus", "daus", 2001, 30000);  System.out.println("Koleksi Buku di Perpustakaan:");  agus.tampilBuku();  }  } |

Hasi RUN



**E. Analisa**

1. Buatlah aplikasi perpustakaan dengan *Constructor* menggunakan Boolean. (*input* pemilihan buku secara dinamis).

|  |
| --- |
| P package studikasus; |

*Script* di atas berfungsi untuk mengelompokkan file *class* yang terkait (karena jenisnya, fungsinya, atau karena alasan lainnya) pada direktori yang sama, dimana didalam setiap *class*nya terdapat *directive* ( *statement* java dalam kode yang di gunakan untuk membuat *class*) *package* yang mengacu pada directori tersebut.

|  |
| --- |
| Impoimport java.util.ArrayList; |

Perintah di atas berfungsi untuk mengimpor *class List*. Dimana *class* *List*  merupakan *class* yang digunakan untuk membuat dan mengelola daftar dinamis dari objek.

|  |
| --- |
| class Buku {  String judul;  String pengarang;  String penerbit;  int tahunTerbit;  double harga; |

Pada *script* di atas terdapat class *buku* yang berfungsi untuk membuat *class* *buku* yang memiliki properti *String judul*, *String Pengarang, String penerbit, int tahunTerbit, dan double harga*.

|  |
| --- |
| public Buku(String j, String p, String pen, int tahun, double h) {  judul = j;  pengarang = p;  penerbit = pen;  tahunTerbit = tahun;  harga = h;  } |

Pada *script* diatas berfungsi untuk membuat *constructor* dengan parameter (String j, String p, String pen, int tahun, dan double h ) yang digunakan untuk mengisi nilai pada masing-masing variabel (judul, pengarang, penerbit, tahunTerbit, dan harga) dengan nilai yang berada pada parameternya.

|  |
| --- |
| *public String toString() {*  *return "Judul Buku = " + judul + ", Pengarang = " + pengarang + ", Penerbit = " + penerbit + ", Tahun Terbit = " + tahunTerbit + ", Harga = " + harga;*  *}*  *}* |

Pada *script* diatas terdapat *public String toString* yang merupakan digunakan untuk mengembalikan representasi teks dari objek. Selanjutnya terdapat ‘public String toString’ ini adalah deklarasi awal dari metode ‘toString’, metode ini akan mengembalikan sebuah String. Ada beberapa property yaitu *return "Judul Buku = " + Judul + ", Pengarang = " + pengarang + ", Penerbit = " + penerbit + ", Tahun Terbit = " + tahunTerbit + ", Harga = " + harga;`*: Ini adalah pernyataan yang mengembalikan sebuah *String* yang berisi informasi tentang objek yang memanggil metode ini. Objek tersebut memiliki atribut-atribut seperti *`judul`, `pengarang`, `penerbit`, `tahun Terbit, dan harga’.*

|  |
| --- |
| *class Perpustakaan {*  *private final ArrayList<Buku> koleksiBuku = new ArrayList<>();* |

Pada *script* diatas terdapat *class Perpustakaan* yang merupakan bahwa kita sedang mendefinisikan sebuah kelas dengan nama “Perpustakaan”. Selanjutnya terdapat *private final ArrayList<Buku> koleksiBuku = new ArrayList<>();* yang berfungsi mendeklarasikan atribut dari kelas “*Perpustakaan*”. Atribut ini adalah sebuah “*ArrayList*” yang berisi objek-objek dari kelas “*Buku*”. Atribut ini dideklarasikan sebagai “*private*”, yang berarti hanya dapat diakses dari dalam kelas “*Perpustakaan*” itu sendiri, dan Atribut yang dideklarasikan sebagai “*final*”, yang berarti setelah insialisasi, referensi ke objek “*ArrayList*” tersebut tidak dapat diubah ke objek lain. Namun, isi dari “*ArrayList*” masih dapat dimodifikasi (yaitu, buku – buku dapat ditambahkan dan dihapus).

|  |
| --- |
| public void tambahBuku(String judul, String pengarang, String penerbit, int tahunTerbit, double harga) |

Pada *script* di atas terdapat perintah public void tambahBuku merupakan *method* utama yang akan dijalankan pertama kali oleh program, `*public void tambahBuku (String judul, String pengarang, String penerbit, int tahunTerbit, double harga)* : Ini adalah deklarasi dari metode *'tambahBuku`*. Metode ini dideklarasikan sebagai `*public*` dan tidak mengembalikan nilai (*void*). Metode ini menerima lima parameter: *`judul`, `pengarang`, `penerbit`, `tahunTerbit`, dan `harga*. Parameter-parameter ini digunakan untuk menginisialisasi objek `*Buku*` yang akan ditambahkan ke koleksi.

|  |
| --- |
| *Buku agus = new Buku(judul, pengarang, penerbit, tahunTerbit, harga);* |

Pada *script* di atas terdapat perintah *Buku agus = new Buku (judul, pengarang, penerbit, tahunTerbit, harga)*; Di dalam metode, terlebih dahulu, objek `*Buku*` baru dengan nama "*agus*" dibuat. Ini dilakukan dengan menggunakan konstruktor `*Buku*` yang menerima nilai-nilai *`judul`, `pengarang`, `penerbit`, `tahunTerbit`, dan `harga* yang diteruskan sebagai parameter. Dengan cara ini, objek `*Buku’* baru diinisialisasi dengan informasi yang sesuai.

|  |
| --- |
| public void tampilBuku() {  for (Buku buku : koleksiBuku) {  System.out.println(buku);  }  }  } |

Pada *script* di atas terdapat *`public void tampilBuku ()* : Ini adalah deklarasi awal dari metode *'tampilBuku*. Metode ini dideklarasikan sebagai `*public*`, yang berarti dapat diakses dari luar kelas, dan tidak mengembalikan nilai (`*void*`). Dengan kata lain, metode ini hanya digunakan untuk menampilkan informasi buku dan tidak mengembalikan nilai kembali. Selanjutnya `*for (Buku buku koleksiBuku)* : Ini adalah *loop `for-each* yang digunakan untuk mengulangi setiap elemen dalam `*koleksiBuku*`, yang diasumsikan sebagai sebuah *ArrayList* yang berisi objek-objek `*Buku*`. Dalam setiap iterasi, objek `*Buku* disimpan dalam variabel ‘*buku*`.

|  |
| --- |
| System.out.println(buku); |

Pada *script* di atas terdapat perintah System.out.print("buku"); yang berfungsi untuk menampilkan Buku di layar.

|  |
| --- |
| public class P4 {  public static void main(String[] args) {  Perpustakaan agus = new Perpustakaan(); |

Pada *script* diatas terdapat public static void main(String[] args) yang merupakan *method* utama yang akan dijalankan pertama kali oleh program, String[] args adalah sebuah *array* dalam bentuk *string*  yang menyimpan *arguments* yang di *passing* melalui *command line* ketika mengeksekusi program. Selanjutnya *`Perpustakaan agus = new Perpustakaan ();*`: Dalam metode `*main*`, Anda membuat objek `*Perpustakaan*` baru dengan nama "*agus*" menggunakan konstruktor yang mungkin ada dalam kelas `*Perpustakaan*. Ini adalah langkah awal dalam membuat objek dari kelas `*Perpustakaan*`.

|  |
| --- |
| agus.tambahBuku("SHolat 5 waktu", "rahmat", "agus", 2004, 40000);  agus.tambahBuku("fisika", "Agus", "daus", 2001, 30000);  System.out.println("Koleksi Buku di Perpustakaan:");  agus.tampilBuku(); |

Pada *script* di atas terdapat *agus.tambahBuku ("fisika", "Agus", "daus", 2001, 30000);`* Ini adalah panggilan metode `*tambahBuku*` pada objek "*agus*". Metode ini digunakan untuk menambahkan buku baru ke koleksi perpustakaan, dan terdapat perintah System.out.print("Koleksi Buku di Perpustakaan:"); yang berfungsi untuk menampilkan Koleksi buku di Perpustakaan di layar. Selanjutnya *agus.tampilBuku();`*: Ini adalah panggilan metode `*tampilBuku*` pada objek "*agus*". Metode ini digunakan untuk menampilkan informasi tentang semua buku dalam koleksi perpustakaan.