Pemrograman Berorientasi Obyek

Polymorphism

Oleh Politeknik Elektronika Negeri Surabaya
2020

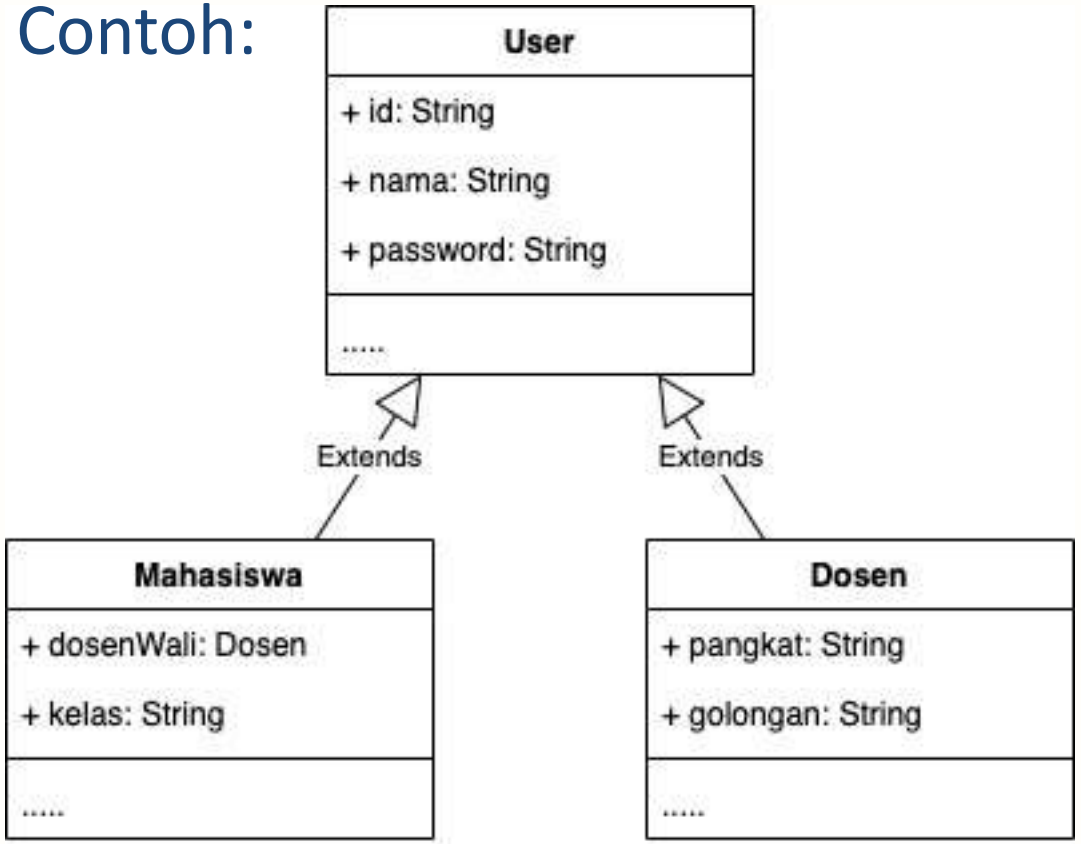
Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Polymorphism

- Polymorphism adalah kemampuan **obyek** untuk mempunyai beberapa **bentuk** class yang **berbeda**.
- Polimorfisme terjadi pada saat suatu obyek bertipe **parent class**, akan tetapi konstruktor yang dipanggil adalah **konstruktor subclass**

Polymorphism

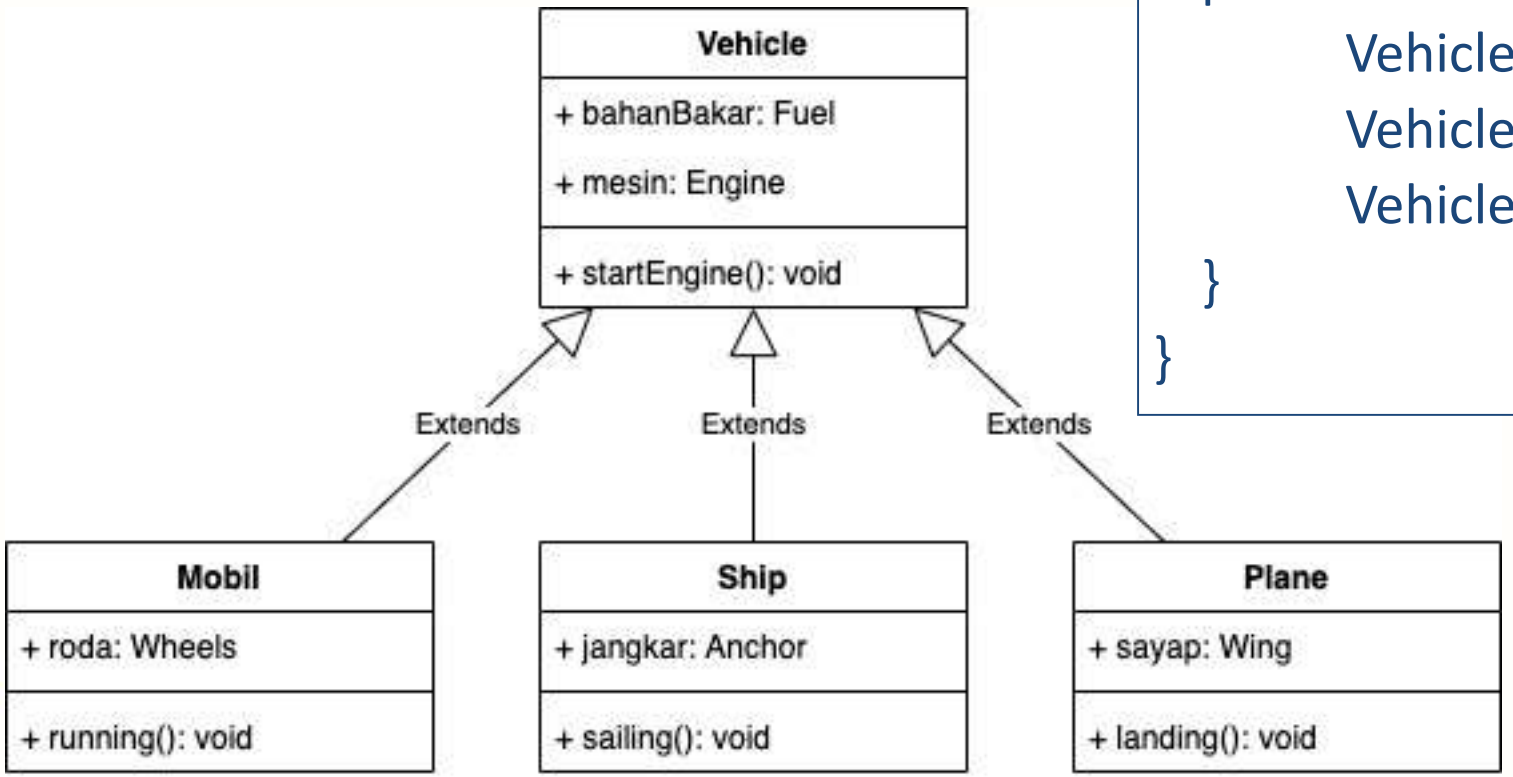
- Contoh:




```
public class Test {
   public static void main(String args[]){
            User fahrul = new Dosen();
            User fadilah = new Mahasiswa();
   }
}
```

Polymorphism

- Contoh:




```
public class Test {
   public static void main(String args[]){
            Vehicle myCar = new Vehicle();
            Vehicle myShip = new Ship();
            Vehicle myPlane = new Plane();
   }
}
```

Virtual Method Invocation

- Virtual method invocation (VMI) merupakan suatu hal yang sangat penting dalam konsep polimorfisme.
- Syarat terjadinya VMI adalah sebelumnya sudah terjadi **polymorphism** dan **overriding**.
- Pada saat obyek yang sudah dibuat tersebut memanggil overridden method pada parent class, kompiler Java akan melakukan invocation (pemanggilan) terhadap overriding method pada subclass, dimana yang seharusnya dipanggil adalah overridden.

Virtual Method Invocation

- Contoh:

```
public class User{
    public void printData(){
        System.out.println("Object User");
    }
}
public class Lecturer extends User{
    public void printData(){
        System.out.println("Object Lecturer");
    }
}
```

```
public class Test {
   public static void main(String args[]){
            User fahrul = new Lecturer();
            fahrul.showData();
   }
}
```

OUTPUT:

Object Lecturer

Virtual Method Invocation

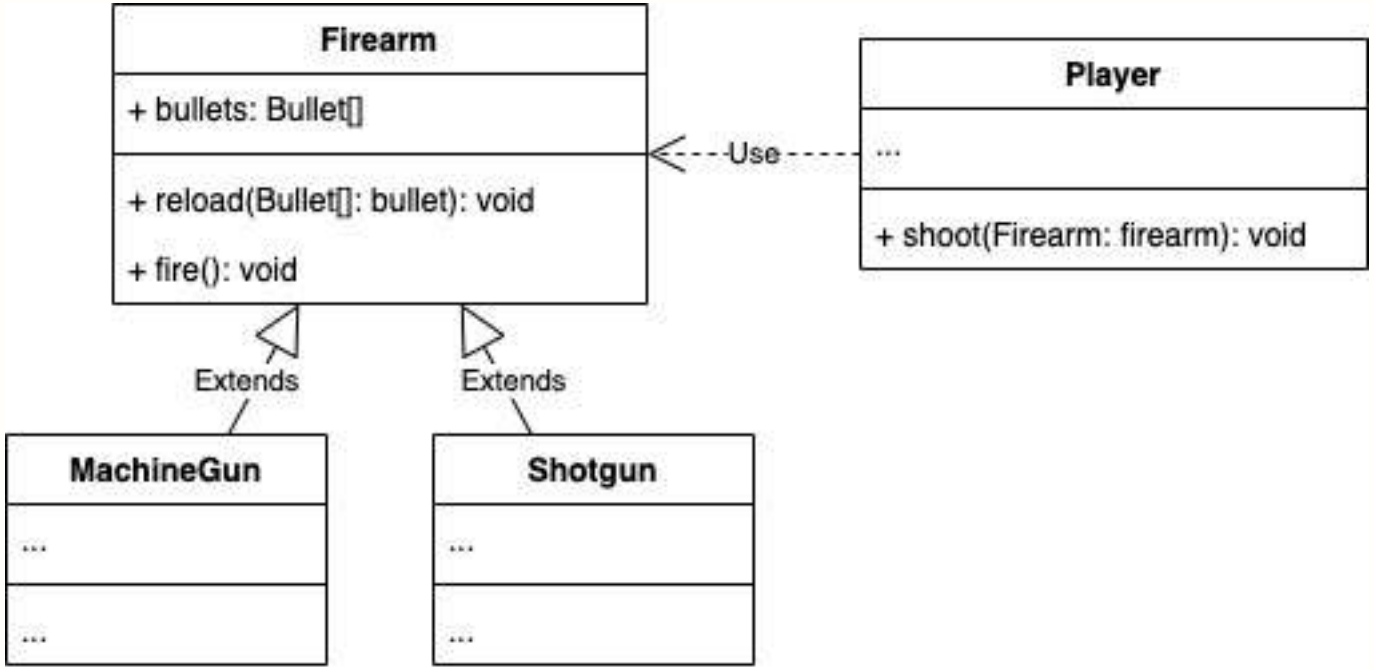
- Bagaimana dengan konstruktor yang dijalankan?
- Pada pembentukan

```
User fahrul = new Lecturer();
```

- Pertama kali akan menjalankan konstruktor Manager, ketika ketemu super() maka akan menjalankan konstruktor Employee (superclass), setelah semua statement dieksekusi baru kemudian melanjutkan menjalankan konstruktor Manager (subclass).

Polymorphic Arguments

Polymorphic arguments adalah tipe data suatu argumen pada suatu method yang bisa menerima suatu nilai yang bertipe subclass-nya.



Polymorphic Arguments

```
public class Firearm{
    public Bullet bullets[];
    public void reload(Bullet bullets[]){…}
    public void fire(){…}
}
```

```
public class MachineGun extends Firearm{
    …
}
public class Shotgun extends Firearm{
    …
}
```

```
public class Player{
    public void shoot(Firearm firearm){…}
}
```

```
public class Test{
    public static void main(String args[]){
        Firearm firearm = new Firearm();
        MachineGun machineGun= new MachineGun();
        Shotgun shotgun= new Shotgun();

        Player firstPerson = nre Player();
        firstPerson.shoot(firearm);
        firstPerson.shoot(machineGun);
        firstPerson.shoot(shotgun);
    }
}
```

Operator **instanceof**

- Pernyataan **instanceof** sangat berguna untuk mengetahui tipe asal dari suatu polymorphic arguments

Operator **instanceof**

```
public class Player{
    public void shoot(Firearm firearm){
        if(firearm instanceof MachineGun){
            //firearm yg digunakan berjenis machinegun
        }else if(firearm instanceof Shotgun){
            //firearm yg digunakan berjenis shotgun
        }else{
            //firearm yg digunakan instance dari firearm
        }
    }
}
```

```
public class Test{
    public static void main(String args[]){
        Firearm firearm = new Firearm();
        MachineGun machineGun= new MachineGun();
        Shotgun shotgun= new Shotgun();

        Player firstPerson = nre Player();
        firstPerson.shoot(firearm);
        firstPerson.shoot(machineGun);
        firstPerson.shoot(shotgun);
    }
}
```

Casting Object

- Seringkali pemakaian instanceof diikuti dengan **casting** object dari tipe parameter ke tipe asal.

Casting Object

```
public class Player{
    public void shoot(Firearm firearm){
        if(firearm instanceof MachineGun){
            MachineGun machinegun = (MachineGun) firearm;
        }else if(firearm instanceof Shotgun){
            Shotgun shotgun = (Shotgun) firearm;
        }else{
            //firearm yg digunakan sudah sbertipe Firearm
        }
    }
}
```

Casting Object

- Tanpa adanya casting obyek, maka nilai yang akan kita pakai setelah proses instanceof masih bertipe parent class-nya, sehingga jika ia perlu dipakai maka ia harus di casting dulu ke tipe subclass-nya.
- Setelah dilakukan casting maka method dan attribute yang dimiliki subclass (class yang dituju) dapat digunakan.

```
...
if (firearm instanceof MachineGun){
    MachineGun machinegun = (MachineGun) firearm;
    //method dan attribute yang dimiliki machinegun telah dapat digunakan
}
...
```

Kenapa diperlukan polymorphic arguments?

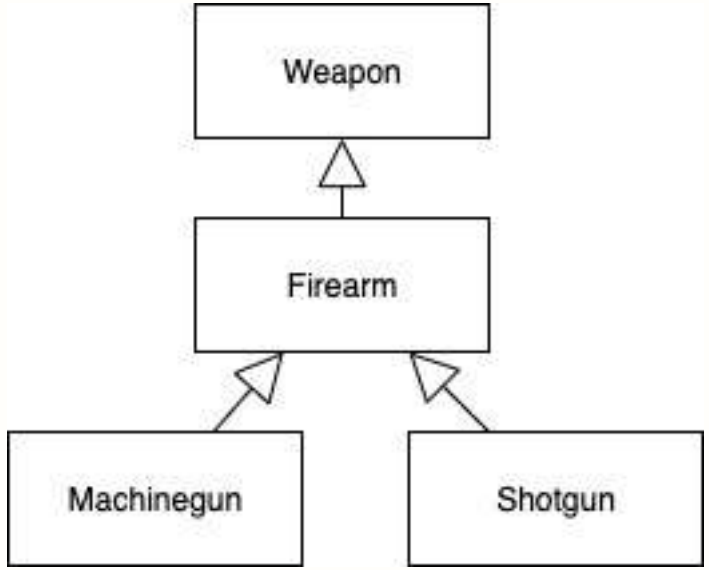
- Mengefisienkan pembuatan program
- Misal Employee mempunyai banyak subclass.
- Maka kita harus mendefinisikan semua method yang menangani behavior dari masing-masing subclass.
- Dengan adanya polymorphic arguments kita cukup mendefinisikan satu method saja yang bisa digunakan untuk menangani behavior semua subclass.

Tanpa polymorphic arguments

```
…
public class Tes {
    public void shootWithMachineGun(MachineGun machineGun) {
                ……
    }
    public static void shootWithShotgun(Shotgun shotgun) {
                ……
    }
     public static void shootWithFirearm(Firearm firearm) {
                ……
    }
}
```

Object Conversion

```
Weapon weapon = new Weapon();
Firearm firearm = new Firearm();
Machinegun machinegun = new Machinegun();
Shotgun shotgun = new Shotgun();
```



weapon = firearm; ✓

Firearm = machinegun; ✓

machinegun = shotgun; ✗

shotgun = firearm; ✗

firearm = weapon; ✗

weapon = machinegun; ✓

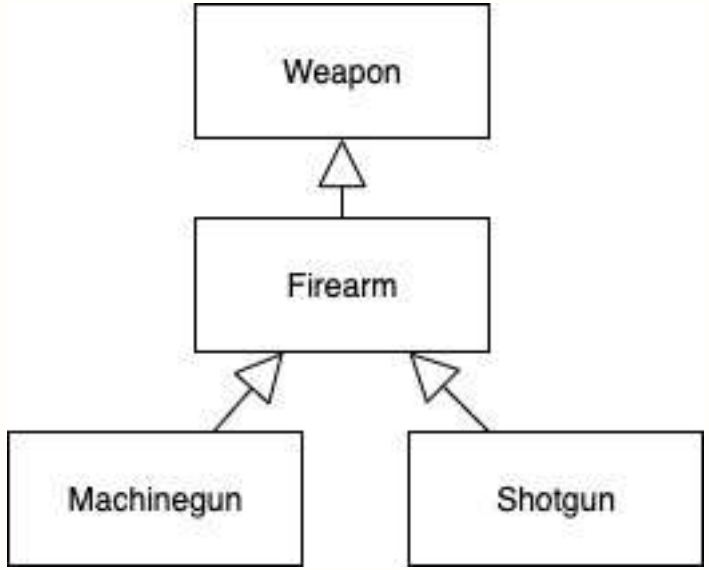
weapon = shotgun; ✓

shotgun = weapon; ✗

machinegun = weapon; ✗

Object Conversion

```
Weapon weapon = new Shotgun();
Firearm firearm = new Machinegun();
Machinegun machinegun = new Machinegun();
Shotgun shotgun = new Shotgun();
```




```
shotgun = weapon;
shotgun = (Shotgun) weapon;
machinegun = firearm;
machinegun = (Machinegun) firearm;
firearm = (Firearm) machinegun;
firearm = (Firearm) shotgun;
```

bridge to the future

http://www.eepis-its.edu

Tugas

1. Apakah yang dimaksud dengan polimorfisme ?
2. Jelaskan proses terjadinya Virtual Method Invocation !
3. Apakah yang dimaksud dengan polymorphic arguments ?
4. Apakah kegunaan kata kunci instanceof ?

1. Oracle Java Documentation, The Java™ Tutorials, https://docs.oracle.com/javase/tutorial/, Copyright © 1995, Oracle 2015.
2. Tita Karlita, Yuliana Setrowati, Rizky Yuniar Hakkun, Pemrograman Berorientasi Obyek, PENS-2012
3. Sun Java Programming, Sun Educational Services, Student Guide, Sun Microsystems, 2001.
4. John R. Hubbard, Programming With Java, McGraw-Hill, ISBN: 0-07-142040-1, 2004.
5. Patrick Niemeyer, Jonathan Knudsen, Learning Java, O'reilly, CA, ISBN: 1565927184, 2000.
6. Philip Heller, Simon Roberts, Complete Java 2 Certification Study Guide, Third Edition, Sybex, San Francisco, London, ISBN: 0-7821-4419-5, 2002.
7. Herbert Schildt, The Complete Reference, Java™ Seventh Edition, Mc Graw Hill, Osborne, ISBN: 978-0-07-163177-8, 2007