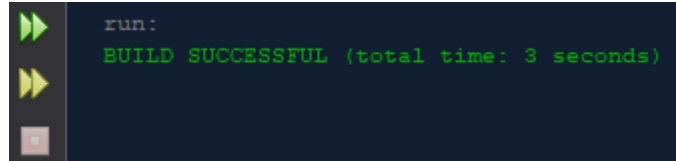




NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

4. PERCOBAAN 1 – BENTUK DASAR POLIMORFISME

4.1. LANGKAH PERCOBAAN



```
run:
BUILD SUCCESSFUL (total time: 3 seconds)
```

4.2. PERTANYAAN

1. Class apa sajakah yang merupakan turunan dari class Employee?

JAWABAN :

Class turunan dari class Employee adalah class PermanentEmployee dan class InternshipEmployee

2. Class apa sajakah yang implements ke interface Payable?

JAWABAN :

Class yang implements ke interface Payable adalah class ElectricityBill dan class PermanentEmployee

3. Perhatikan class Tester1, baris ke-10 dan 11. Mengapa e, bisa diisi dengan objek pEmp (merupakan objek dari class PermanentEmployee) dan objek iEmp (merupakan objek dari class InternshipEmployee) ?

JAWABAN :

Karena objek pEmp (dibuat dari class PermanentEmployee) dan objek iEmp (dibuat dari class InternshipEmployee) dan kedua class itu berasal dari class turunan / class yang meng-*extend* dari superclass yaitu class Employee.

4. Perhatikan class Tester1, baris ke-12 dan 13. Mengapa p, bisa diisi dengan objek pEmp (merupakan objek dari class PermanentEmployee) dan objek eBill (merupakan objek dari class ElectricityBill) ?

JAWABAN :

Karena objek pEmp (dibuat dari class PermanentEmployee) dan objek eBill (dibuat dari class ElectricityBill) dan kedua class itu adalah class yang meng-*implement* interface IPayable



NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

5. Coba tambahkan sintaks: `p = iEmp`; `e = eBill`; pada baris 14 dan 15 (baris terakhir dalam method main) !
Apa yang menyebabkan error?

JAWABAN :

```
12 public class Tester1 {  
13     public static void main(String[] args) {  
14         PermanentEmployee pEmp = new PermanentEmployee("Dedik", 500);  
15         InternshipEmployee iEmp = new InternshipEmployee("Sunarto", 5);  
16         ElectricityBill eBill = new ElectricityBill(5, "A-1");  
17         Employee e;  
18         Payable p;  
19         e = pEmp;  
20         e = iEmp;  
21         p = pEmp;  
22         p = eBill;  
23         p = iEmp;  
24         e = eBill;  
25     }
```

Terjadi ERROR, dikarenakan

- objek `iEmp` yang berasal dari class `InternshipEmployee`, dan class `InternshipEmployee` nya tidak meng-*implement* interface `IPayable` dan
- objek `eBill` yang berasal dari class `ElectricityBill`, dan class `ElectricityBill` nya tidak meng-*extend* class `Employee`

6. Ambil kesimpulan tentang konsep/bentuk dasar polimorfisme!

JAWABAN :

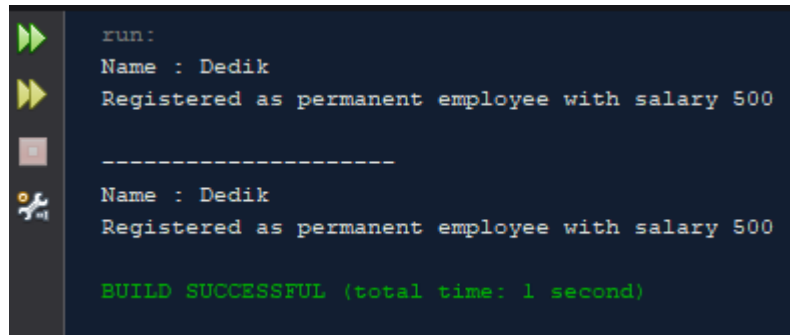
Kesimpulan nya : Ketika ada suatu objek yang dideklarasikan dari super class, maka objek tersebut bisa diinstansiasi sebagai objek dari sub class. Dari itu bisa kita lihat bahwa konsep polimorfisme bisa diterapkan pada class-class yang memiliki relasi inheritance.



NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

5. PERCOBAAN 2 – VIRTUAL METHOD INVOCATION

5.1. LANGKAH PERCOBAAN



```
run:
Name : Dedik
Registered as permanent employee with salary 500

-----
Name : Dedik
Registered as permanent employee with salary 500

BUILD SUCCESSFUL (total time: 1 second)
```

5.2. PERTANYAAN

1. Perhatikan class Tester2 di atas, mengapa pemanggilan `e.getEmployeeInfo()` pada baris 8 dan `pEmp.getEmployeeInfo()` pada baris 10 menghasilkan hasil sama?

JAWABAN :

Karena pada objek `e` dan objek `pEmp` itu memanggil method sama yaitu method `getEmployeeInfo()`

```
System.out.println(""+e.getEmployeeInfo());
System.out.println("-----");
System.out.println(""+pEmp.getEmployeeInfo());
```

Dan pada objek `e` dan objek `pEmp` sudah di deklarasikan sebelumnya pada baris ke-14 (gambar dibawah) jadi `e = pEmp` itu class `Employee` dan bisa mengakses semua variabel dan method yang ada pada class `PermanentEmployee`.

```
PermanentEmployee pEmp = new PermanentEmployee("Dedik",500);
Employee e;
e = pEmp;
```

2. Mengapa pemanggilan method `e.getEmployeeInfo()` disebut sebagai pemanggilan method virtual (virtual method invocation), sedangkan `pEmp.getEmployeeInfo()` tidak?

JAWABAN :

Karena objek `pEmp` dideklarasikan langsung dari class `PermanentEmployee` maka ketika memanggil method `pEmp.getEmployeeInfo()` itu bukan method virtual. Dan untuk objek `e` itu dari class `Employee`, oleh karena itu jika pemanggilan method `e.getEmployeeInfo()` maka disebut sebagai pemanggilan method virtual karena tidak langsung dideklarasikan dari class `PermanentEmployee`



NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

3. Jadi apakah yang dimaksud dari virtual method invocation? Mengapa disebut virtual?

JAWABAN :

Virtual Method Incovation (VMI) adalah method yang terjadi pada polimorfisme dan overriding. Jadi pada saat objek yang sudah dibuat tersebut memanggil override method pada superclass, maka compiler Java akan melakukan pemanggilan terhadap overriding method pada subclass, dimana yang seharusnya dipanggil adalah override method.

6. PERCOBAAN 3 – HETEROGENOUS COLLECTION

6.1. LANGKAH PERCOBAAN

```
run:  
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.2. PERTANYAAN

1. Perhatikan array e pada baris ke-8, mengapa ia bisa diisi dengan objekobjek dengan tipe yang berbeda, yaitu objek pEmp (objek dari PermanentEmployee) dan objek iEmp (objek dari InternshipEmployee) ?

JAWABAN :

Hal ini bisa dilakukan karena array e dideklarasikan dari class Employee dan class Employee adalah superclass dari PermanentEmployee dan IntershipEmployee.

Oleh karena itu array e bisa diisi dengan objek objek dengan tipe yang berbeda.

2. Perhatikan juga baris ke-9, mengapa array p juga biisi dengan objek-objek dengan tipe yang berbeda, yaitu objek pEmp (objek dari PermanentEmployee) dan objek eBill (objek dari ElectricityBilling) ?

JAWABAN :

Hal ini bisa dilakukan karena array p dideklarasikan dari interface IPayable dan objek pEmp berasal dari class PermanentEmployee juga objek eBill berasal dari class ElectricityBill.

➔ Jadi class PermanentEmployee dan class ElectricityBill meng-*implement* dari interface IPayable.

Oleh karena itu array p bisa diisi dengan objek objek dengan tipe yang berbeda.



NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

3. Perhatikan baris ke-10, mengapa terjadi error?

JAWABAN :

Terjadi error karena objek eBill dideklarasikan dari class ElectricityBill dan class ElectricityBill bukan superclass dari class Employee (tidak extends ke class Employee).

7. PERCOBAAN 4 – ARGUMEN POLIMORFISME, INSTANCEOF DAN CASTING OBJEK

7.1. LANGKAH PERCOBAAN

```
run:
Total payment = 500
kWH = 5
Category = R-1(100 per kWH)

-----
Total payment = 525
Name : Dedik
Registered as permanent employee with salary 500

-----
Name : Dedik
Registered as permanent employee with salary 500

You have to pay her/him monthly!!!
-----
Name : Sunarto
Registered as intership employee for 5 month/s

No need to pay him/her :)
BUILD SUCCESSFUL (total time: 1 second)
```

7.2. PERTANYAAN

1. Perhatikan class Tester4 baris ke-7 dan baris ke-11, mengapa pemanggilan ow.pay(eBill) dan ow.pay(pEmp) bisa dilakukan, padahal jika diperhatikan method pay() yang ada di dalam class Owner memiliki argument/parameter bertipe Payable? Jika diperhatikan lebih detil eBill merupakan objek dari ElectricityBill dan pEmp merupakan objek dari PermanentEmployee?

JAWABAN :

Karena pemanggilan method ow.pay(eBill) dan ow.pay(pEmp) bisa dilakukan itu karena class ElectricityBill dan class PermanentEmployee sudah meng-implement interface IPayable.



NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

2. Jadi apakah tujuan membuat argument bertipe Payable pada method pay() yang ada di dalam class Owner?

JAWABAN :

Tujuan membuat argument bertipe Payable pada method pay() di class Owner agar Owner bisa melakukan pembayaran baik kepada karyawan permanen / tetap maupun rekening listrik melalui method pay() yang didapat dengan membuat argumen bertipe IPayable (interface IPayable).

3. Coba pada baris terakhir method main() yang ada di dalam class Tester4 ditambahkan perintah ow.pay(iEmp); Mengapa terjadi error?

```
3 public class Tester4 {
4     public static void main(String[] args) {
5         Owner ow = new Owner();
6         ElectricityBill eBill = new ElectricityBill(5, "R-1");
7         ow.pay(eBill); //pay for electricity bill
8         System.out.println("-----");
9
10        PermanentEmployee pEmp = new PermanentEmployee("Dedik", 500);
11        ow.pay(pEmp); //pay for permanent employee
12        System.out.println("-----");
13
14        InternshipEmployee iEmp = new InternshipEmployee("Sunarto", 5);
15        ow.showMyEmployee(pEmp); //show permanent employee info
16        System.out.println("-----");
17        ow.showMyEmployee(iEmp); //show internship employee info
18
19        ow.pay(iEmp);
20    }
21 }
```

JAWABAN :

Sudah dicoba dan terjadi Error dikarenakan class InternshipEmployee tidak melakukan implement pada interface Payable (class IPayable)

```
public class InternshipEmployee extends Employee{
```

```
12 public class Tester4{
13     public static void main(String[] args) {
14         Owner ow = new Owner();
15         ElectricityBill eBill = new ElectricityBill(5, "R-1");
16         ow.pay(eBill);
17         System.out.println("-----");
18
19         PermanentEmployee pEmp = new PermanentEmployee("Dedik", 500);
20         ow.pay(pEmp);
21         System.out.println("-----");
22
23         InternshipEmployee iEmp = new InternshipEmployee("Sunarto", 5);
24         ow.showMyEmployee(pEmp);
25         System.out.println("-----");
26         ow.showMyEmployee(iEmp);
27
28         ow.pay(iEmp);
29     }
30 }
```



NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

4. Perhatikan class Owner, diperlukan untuk apakah sintaks `p instanceof ElectricityBill` pada baris ke-6 ?

JAWABAN :

Sintaks *instanceof* digunakan untuk mengecek apakah suatu objek merupakan hasil instansiasi dari suatu class tertentu atau tidak. Hasil dari *instanceof* `ElectricityBill` berupa *boolean*. Seperti gambar ini:

```
if(p instanceof ElectricityBill){
    ElectricityBill eb = (ElectricityBill) p;
    System.out.println(""+eb.getBillInfo());
}else if(p instanceof PermanentEmployee){
    PermanentEmployee pe = (PermanentEmployee) p;
    pe.getEmployeeInfo();
    System.out.println(""+pe.getEmployeeInfo());
}
```

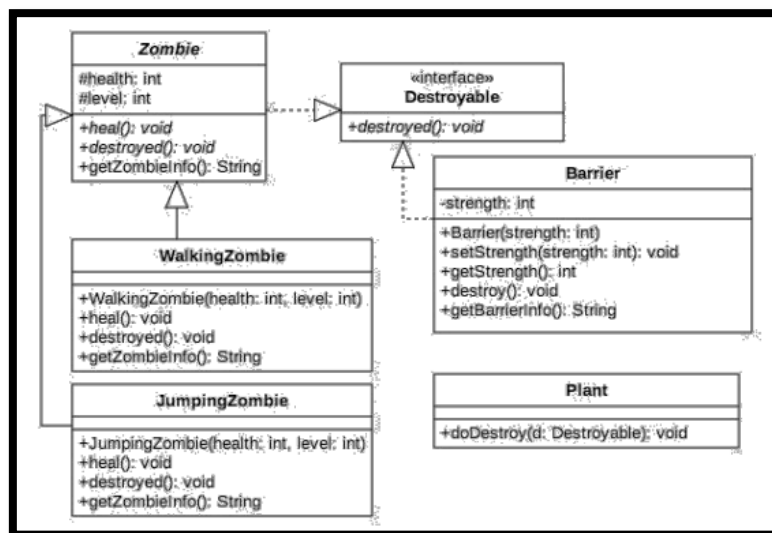
5. Perhatikan kembali class Owner baris ke-7, untuk apakah casting objek disana (`ElectricityBill eb = (ElectricityBill) p`) diperlukan ? Mengapa objek `p` yang bertipe `Payable` harus di-casting ke dalam objek `eb` yang bertipe `ElectricityBill` ?

JAWABAN :

Tujuan dari *casting* objek pada `ElectricityBill eb = (ElectricityBill) p` digunakan untuk mengubah tipe dari suatu objek pada `ElectricityBill`. Proses ini sering disebut sebagai *explicit casting*, karena bentuk tujuan dari casting harus dituliskan dalam tanda kurung, sehingga di depan objek yang akan di-*casting*.

8. TUGAS

Buat program dari class diagram di bawah ini





NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

JAWABAN TUGAS :

Terdapat 7 class :

1. Class Zombie

```
13 public class Zombie implements IDestroyable{
14     protected int health;
15     protected int level;
16
17     public void heal() {
18
19     }
20
21     public void destroyed() {
22
23     }
24
25     public String getZombieInfo() {
26         return "Zombie Data = ";
27     }
28 }
```

2. Class WalkingZombie

```
12 public class WalkingZombie extends Zombie{
13     public WalkingZombie(int health, int level){
14         super.health = health;
15         super.level = level;
16     }
17
18     @Override
19     public void heal() {
20         if (level == 1){
21             health += (health * 10/100);
22         } else if (level == 2){
23             health += (health * 30/100);
24         } else if (level == 3){
25             health += (health * 40/100);
26         }
27     }
28
29     @Override
30     public void destroyed() {
31         health -= (health * 20/100);
32     }
33
34     @Override
35     public String getZombieInfo() {
36         String info = "Walking " + super.getZombieInfo() + "\n";
37         info += "Health = " + health + "\n" + "Level = " + level + "\n";
38         return info;
39     }
40 }
```




NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

3. Class JumpingZombie

```
12 public class JumpingZombie extends Zombie{
13     public JumpingZombie(int health, int level){
14         super.health = health;
15         super.level = level;
16     }
17     @Override
18     public void heal(){
19         if(level == 1){
20             health += (health * 30/100);
21         }else if(level == 2){
22             health += (health * 40/100);
23         }else if(level == 3){
24             health += (health * 50/100);
25         }
26     }
27     @Override
28     public void destroyed(){
29         health -= (health * 10/100);
30     }
31     @Override
32     public String getZombieInfo(){
33         String info = "Jumping "+super.getZombieInfo() +"\n";
34         info += "Health = "+health+"\n"+"Level = "+level+"\n";
35         return info;
36     }
37 }
```

4. Class IDestroyable (interface)

```
1 public interface IDestroyable {
2     public abstract void destroyed();
3 }
14 }
```

5. Class Plant

```
12 public class Plant {
13     public void doDestroy(IDestroyable d){
14         d.destroyed();
15     }
16 }
```



NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

6. Class Barrier

```
12 public class Barrier implements IDestroyable{
13     private int strength;
14
15     public Barrier(int strength) {
16         this.strength = strength;
17     }
18
19     public void setStrength(int strength){
20         this.strength = strength;
21     }
22
23     public int getStrength() {
24         return strength;
25     }
26
27     public void destroyed(){
28         strength -= (strength * 0.1);
29     }
30
31     public String getBarrierInfo(){
32         String info = "";
33         info += "Barrier Strength = "+strength;
34         return info;
35     }
36 }
```

7. Class TesterTugas (Main Program)

```
12 public class TesterTugas {
13     public static void main(String[] args) {
14         WalkingZombie wz = new WalkingZombie(100,1);
15         JumpingZombie jz = new JumpingZombie(100,2);
16         Barrier b = new Barrier(100);
17         Plant p = new Plant();
18         System.out.println(" "+wz.getZombieInfo());
19         System.out.println(" "+jz.getZombieInfo());
20         System.out.println(" "+b.getBarrierInfo());
21         System.out.println("=====");
22         for(int i=0; i<4; i++){
23             p.doDestroy(wz);
24             p.doDestroy(jz);
25             p.doDestroy(b);
26         }
27         System.out.println(" "+wz.getZombieInfo());
28         System.out.println(" "+jz.getZombieInfo());
29         System.out.println(" "+b.getBarrierInfo());
30     }
31 }
```



NAMA : ANA QONITAH MUNAWWAROH
NIM : 2041720118
KELAS : 2C / 03
MATERI : LAPORAN PRAKTIKUM JOBSHEET 11 PRAKPBO

HASIL OUTPUT KODE PROGRAM

```
run:
Walking Zombie Data =
Health = 100
Level = 1

Jumping Zombie Data =
Health = 100
Level = 2

Barrier Strength = 100
=====
Walking Zombie Data =
Health = 42
Level = 1

Jumping Zombie Data =
Health = 66
Level = 2

Barrier Strength = 64
BUILD SUCCESSFUL (total time: 0 seconds)
```