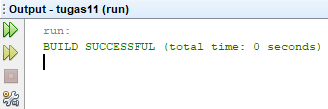
1. Output Percobaan 1



Jawaban pertanyaan

1. Class InternshipEmployee dan class PermanentEmployee

2. Class PermanentEmployee dan class ElectricityBill

3. karena objek e adalah Employee lalu objek pEmp dan iEmp adalah class PermanentEmployee dan class InternshipEmployee dimana class tersebut turunan dari class Employee

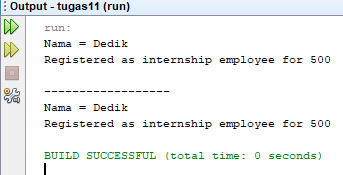
4. karena objek p adalah payable lalu objek pEmp dan eBill adalah class PermanentEmployee dan class ElectricityBill dimana class tersebut implement dari class interface Payable

5. p = iEmp; error karena iEmp bukan termasuk implement dari p / payable

e = eBill; error karena eBill bukan termasuk turunan dari e / Employee

6. fungsi implement digunakan jika parent tersebut merupakan interface, jika bukan merupakan interface menggunakan fungsi turunan/extends

1. Output Percobaan 2



Jawaban pertanyaan

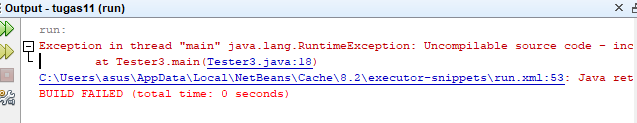
1.karena terdapat indeks program yaitu

, yang menyebabkan hasil dari 2 class tersebut sama

2. karena objek polimorfisme e/employee kemudian dipanggil method overriding yaitu **getEmployeeInfo()**

3. Virtual method invocation terjadi ketika ada pemanggilan overriding method dari suatu objek polimorfisme. Disebut virtual karena antara method yang dikenali oleh compiler dan method yang dijalankan oleh JVM berbeda

1. Output Percobaan 3



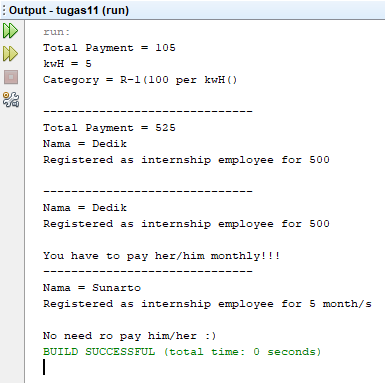
Jawaban pertanyaan

1. karena objek pEmp (objek dari PermanentEmployee) dan objek iEmp (objek dari InternshipEmployee) merupakan turunan dari objek class e/Employee

2. karena objek pEmp (objek dari PermanentEmployee) dan objek eBill (objek dari ElectricityBilling) merupakan implement dari objek class p/Payable

3. karena terdapat objek eBill (objek dari ElectricityBilling) yang dimana objek tersebut bukan merupakan turunan dari dari objek class e/Employee

1. Output Percobaan 4



Jawaban pertanyaan

1. karena objek pEmp (objek dari PermanentEmployee) dan objek eBill (objek dari ElectricityBilling) merupakan implement dari objek class p/Payable

2. tujuannya untuk menghitung total payment yang dimana didalamnya terdapat objek pEmp (objek dari PermanentEmployee) dan objek eBill (objek dari ElectricityBilling)

3.Terjadi error karena terdapat method pay() yang ada di dalam class Ownermemiliki argument/parameter bertipe Payable kemudian memamnggil iEmp (objek dari InternshipEmployee) yang dimana bukan merupakan implement dari objek class p/Payable

4. yaitu untuk mengecek apakah objek **ElectricityBill** merupakan hasil instansiasi dari class Payable

5. karena casting objek tersebut berguna untuk mengubah tipe dari objek p/Payable **dari objek ElectricityBill**

1. Tugas

Kode Program

1. Class interface Destroyable

package zombie;

/\*\*

\*

\* @author asus

\*/

public interface IDestroyable {

public abstract void destroyed();

}

1. Class Zombie

package zombie;

/\*\*

\*

\* @author asus

\*/

public abstract class Zombie implements IDestroyable{

protected int health,level;

public abstract void heal();

@Override

public abstract void destroyed();

public String getZombieInfo(){

return "Health = "+health+"\nLevel = "+level+"\n";

}

}

1. Class Barrie

package zombie;

/\*\*

\*

\* @author asus

\*/

public class Barrier implements IDestroyable{

private int strength;

public Barrier(int strength) {

this.strength = strength;

}

public int getStrength() {

return strength;

}

public void setStrength(int strength) {

this.strength = strength;

}

@Override

public void destroyed() {

strength\*=0.9;

}

public String getBarrierInfo(){

return "Barrier Strength = "+strength+"\n";

}

}

1. Class WalkingZombie

package zombie;

/\*\*

\*

\* @author asus

\*/

public class WalkingZombie extends Zombie{

public WalkingZombie(int health,int level) {

this.health=health;

this.level=level;

}

@Override

public void heal() {

switch(level){

case 1:health\*=1.1;break;

case 2:health\*=1.3;break;

case 3:health\*=1.4;break;

}

}

@Override

public void destroyed() {

health-=health\*20/100;

}

public String getZombieInfo(){

return "Walking Zombie Data =\n"+super.getZombieInfo();

}

}

1. Class JumpingZombie

package zombie;

/\*\*

\*

\* @author asus

\*/

public class JumpingZombie extends Zombie{

public JumpingZombie(int health,int level) {

this.health=health;

this.level=level;

}

@Override

public void heal() {

switch(level){

case 1:health\*=1.3;break;

case 2:health\*=1.4;break;

case 3:health\*=1.5;break;

}

}

@Override

public void destroyed() {

health-=health\*10/100;

}

public String getZombieInfo(){

return "Jumping Zombie Data =\n"+super.getZombieInfo();

}

}

1. Class Plant

package zombie;

/\*\*

\*

\* @author asus

\*/

public class Plant {

public void doDestroy(IDestroyable d){

if(d instanceof WalkingZombie){

((WalkingZombie) d).destroyed();

}else if(d instanceof JumpingZombie){

((JumpingZombie) d).destroyed();

}else if(d instanceof Barrier){

((Barrier)d).destroyed();

}

}

}

1. Class Tester

package zombie;

/\*\*

\*

\* @author asus

\*/

public class Tester {

public static void main(String[] args) {

WalkingZombie wz=new WalkingZombie(100,1);

JumpingZombie jz=new JumpingZombie(100,2);

Barrier b=new Barrier(100);

Plant p=new Plant();

System.out.println(""+wz.getZombieInfo());

System.out.println(""+jz.getZombieInfo());

System.out.println(""+b.getBarrierInfo());

System.out.println("----------------------");

for(int i=0;i<4;i++){

p.doDestroy(wz);

p.doDestroy(jz);

p.doDestroy(b);

}

System.out.println(""+wz.getZombieInfo());

System.out.println(""+jz.getZombieInfo());

System.out.println(""+b.getBarrierInfo());

}

}

Outputnya:

