

JOB SHEET 2
CLASS AND
OBJECT



From:

AL AZHAR RIZQI RIFA'I FIRDAUS

Class:

11

Absence:

01

Major:

Information Technology

Study Program:

Informatic
Engineering

1. COMPETENCIES

- a. Students understand the most basic concept of object, class and object
- b. Students are able to declare class, attribute and method
- c. Students are able to create object (instantiation)
- d. Students are able to access attribute and method of object
- e. Students are able to implement constructor

2. OVERVIEW

Generally, there are 2 paradigms in developing programs. They are Procedural Programming and Object Oriented Programming. In Procedural Programming, the program is organized as a collection of functions or subprocesses, while Object Oriented Programming (OOP) composes program by collection of objects and there are interactions between objects. Object becomes main concern in OOP. Because of the importance of object in OOP, understanding the context of object becomes very principal in developing the program. The OOP concept translates real life into programming concepts.

In the real life, **OBJECT** is a real object that could be found around us. It means that the object must be a real thing. Something that is not real, just a prototype/design/concept could be classified as object. If we see more detail on an object, we will find that it contains of 2 main characteristics, it has something and it can do something. Something owned by an object called as **attribute**/property/data/character/state/field. And something that could be performed by an object called as **method**/behaviour/function/procedure.

Every object must come from a design/blueprint. The design will define the template from which the object will be created. Design/prototype/template/blueprint from which objects are created know as **CLASS**. It means that the object is the instance of a class. It is a real thing created from class. The process creating an object from a class known as **INSTANTIATION**. From here, we can conclude that, there is no object that exists without a class from which it is created. We need to define a class before creating an object. After creating an object, we can manipulate it. We could not do any manipulation to a class instead, because it is still a design/concept.

Declaring class in source code is so simple. Keyword **class** takes place on it. Below is the format of class declaration.

```
class ClassName {  
    // attribute  
    // method  
}
```

After declaring a class, we need to declare attributes and methods owned by this class. Attributes depict data/property owned by class, while methods depict process that could be performed by class.

Below is the format of attribute declaration:

```
dataType attributeName;
```

While below is the format of method declaration:

```
dataType methodName (dataType parameter){  
    // method body  
}
```

Note:

Data type of method : it determines the return value of the method. If we need the method not to return any value, then we can give it void datatype. But if we need the method to return a value, then we must declare it with nonDvoid datatype. It could be int if we need the method to return int value, String data type for a method with String return value, etc.

Parameter of method : it is used to pass the value from outside of method to be processed inside the method. If we need to be able to pass values from outside of method when we call the method, then we need to make the method to have parameters. Of course the values that will be passed through parameters will be processed inside method.

Example:

```
class Bicycle {  
    double speed;  
    int gear;  
  
    void changeGear(int  
        g){gear = g;  
    }  
    void speedUp(double  
        v){speed = speed+v;  
    }  
    void applyBrakes(double  
        v){speed = speed-v;  
    }  
    void printStates(){ System.out.println("Speed  
        = "+speed);System.out.println("Gear = "+gear);  
    }  
}
```

Class that is already created, could not be used as long as we does not create an object from that class. Thus, the next step after creating a class is creating an object from the class (instantiation). Instantiation could be written as follows:

```
ClassNameObjectName= newClassName();
```

Example:

```
Bicycleb= new Bicycle();
```

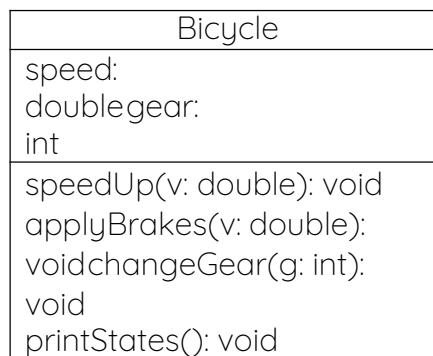
Once we have finished to create an object, further we can use or manipulate it. For example, we can assign value to its attributes, we can call its method as well. Below is the way how we can access attribute of the object:

```
b.speed = 10;
b.gear = 1;
```

And below is the way to call method of the

```
object:b.speedUp(5);
b.changeGear(1); b.printStates();
```

The design/structure of a class could be visualized by using **CLASS DIAGRAM**. The following image shows the example of class diagram of class Bicycle.



When creating an object from a class (instantiation), if we look it closer, we will find that actually there is a special method that runs to create the object. A special method that runs at the instantiation process that creates an object is called as **CONSTRUCTOR**. The characteristics of constructor are:

- Constructor name is similar with the class name

- Constructor does not have any data type

- Constructor will never need a **return** statement

- Constructor only can be run at the instantiation

process The following syntax will give the format to declare constructor.

```
NamaClass(tipeData parameter){
}
```

Example:

```
Bicycle(){
    // constructor body
}
Bicycle(int g, double v){
    // constructor body
}
```

There are some naming conventions that we need to care about:

- Class Name : Class name must be a noun, and it is written in Capital Camel Case. For example: Bicycle, InternationalStudent

- Attribut Name : Attribute name must be a noun, and it is

written in Camel Case. Example: name, totalAmount

Method name : method name must be a verb and it is written in Camel Case. Example: print(), calculateDiscount()

3. LABS ACTIVITY

PART 1 – NetBeans Installation

NetBeans is integrated development environment that could be used to manage Java program. By using NetBeans, writing Java program becomes easier (because of syntax completion and javadoc feature) and becomes less of typo. The current latest version of NetBeans is version 10.

1. Your computer must install JDK first, before installing NetBeans. The JD installer could be found at <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.
2. Download NetBeans installer at <https://netbeans.org/downloads/>
3. Run the installer, and follow the installation steps
4. Run the NetBeans
5. Create a new Project, **File > New Project**. In the **Categories** panel choose **Java** and in **Projects** panel choose **Java Application**. Then give a **Project Name** (for example **AlgorithmAndDataStructure**), **Project Location** and **Project Folder**, and uncheck **Create Main Class** option.
6. Press the **Finish** button. The new project will be created and this project that will be used by us to store all code will be developed this semester for this course
7. Right click at **Sources Packages > New > Java Package**. Type **minggu1** as the package name. Then, a new package named **minggu1** will be created.
8. Right click at **minggu1 > New > Java Class**. Type **HelloNetBeans** as the class name. And in the editor **HelloNetBeans**, complete the main method as follows:

```
1 package minggu1;
2
3 public class HelloNetBeans {
4     public static void main(String[] args) {
5         System.out.println("Hello NetBeans");
6     }
7 }
```

9. Right click at class **HelloNetBeans** > **Compile** to compile this source code
10. Right click at class **HelloNetBeans** > **Run File** to run this file

PART 2 – Declaring Class, Attribute and Method

Baran
namaBarang:
String jenisBarang:
String stok: int

hargaSatuan: int
tampilBarang(): void
tambahStok(n: int): void
kurangiStok(n: int): void
hitungHargaTotal(jumlah: int):

Based on the class diagram above, we will create a java program.

1. We have created a new project at PART 1. We will still use it
2. In the package `minggu1`, create a new class named `Barang`.
3. Based on the class diagram above, complete the class declaration by declaring the correct attributes and methods:

```

1  package minggu1;
2
3  public class Barang {
4      String namaBarang, jenisBarang;
5      int stok, hargaSatuan;
6
7      void tampilBarang(){
8          System.out.println("Nama = "+namaBarang);
9          System.out.println("Jenis = "+jenisBarang);
10         System.out.println("Stok = "+stok);
11         System.out.println("Harga Satuan = "+hargaSatuan);
12     }
13     void tambahStok(int n){
14         stok = stok+n;
15     }
16     void kurangiStok(int n){
17         stok = stok-n;
18     }
19     int hitungHargaTotal(int jumlah){
20         return jumlah*hargaSatuan;
21     }
22 }

```

4. Try to Run class/file `Barang`. Does it works?

PART 3 – Instantiating Object and Accessing Atribut&Method of the Object

1. We have created class `Barang`. And to start using it, we need to createan object from it. After creating the object, of course we will be able toaccess its attributes and methods.
2. In the package `minggu1`, create a new class named `BarangMain`. Then, create a `main` method in the class `BarangMain`
3. In the method `main()`, do the instantiation, and then access the attributes and methods of the created object


```

1  package minggu1;
2
3  public class BarangMain {
4      public static void main(String[] args) {
5          Barang b1 = new Barang();
6          b1.namaBarang = "Corsair 2 GB";
7          b1.jenisBarang = "DDR";
8          b1.hargaSatuan = 250000;
9          b1.stok = 10;
10         b1.tambahStok(1);
11         b1.kurangiStok(3);
12         b1.tampilBarang();
13         int hargaTotal = b1.hitungHargaTotal(4);
14         System.out.println("Harga 4 buah = "+hargaTotal);
15     }
16 }

```

4. Run class `BarangMain` and observe the result.

PART 4 – Constructor

1. Look again at class `Barang`. Add in the class `Barang` 2

```

1  package minggu1;
2
3  public class Barang {
4      String namaBarang, jenisBarang;
5      int stok, hargaSatuan;
6
7      Barang(){
8      }
9      Barang(String nm, String jn, int st, int hs){
10         namaBarang = nm;
11         jenisBarang = jn;
12         stok = st;
13         hargaSatuan = hs;
14     }
15
16     void tampilBarang(){
17         System.out.println("Nama = "+namaBarang);
18         System.out.println("Jenis = "+jenisBarang);
19         System.out.println("Stok = "+stok);
20         System.out.println("Harga Satuan = "+hargaSatuan);
21     }
22     void tambahStok(int n){
23         stok = stok+n;
24     }
25     void kurangiStok(int n){
26         stok = stok-n;
27     }
28     int hitungHargaTotal(int jumlah){
29         return jumlah*hargaSatuan;
30     }
31 }

```

2. Now move to class `BarangMain`. Create one more object, but

now we will use parametric constructor at the instantiation.

```

1  package minggu1;
2
3  public class BarangMain {
4      public static void main(String[] args) {
5          Barang b1 = new Barang();
6          b1.namaBarang = "Corsair 2 GB";
7          b1.jenisBarang = "DDR";
8          b1.hargaSatuan = 250000;
9          b1.stok = 10;
10         b1.tambahStok(1);
11         b1.kurangiStok(3);
12         b1.tampilBarang();
13         int hargaTotal = b1.hitungHargaTotal(4);
14         System.out.println("Harga 4 buah = "+hargaTotal);
15
16         Barang b2 = new Barang("Logitech", "Wireless Mouse",
17                                150000, 25);
18         b2.tampilBarang();
19     }
20 }

```

3. Run the class **BarangMain** and observe the result.

11. QUESTION

1. Mention 2 characteristics of class/object!

1. Encapsulation: Classes and objects in object-oriented programming (OOP) typically exhibit encapsulation, which means that the internal workings of an object are hidden from the outside world. This is achieved by defining public methods and properties that can be accessed by other parts of the program, while keeping the implementation details private. Encapsulation is important for creating modular, maintainable code.

2. Inheritance: Inheritance is another characteristic of OOP, whereby a class can inherit properties and methods from a parent class. This allows for code reuse and helps to create a hierarchical class structure. For example, you might have a base class called "Animal", which contains common properties and methods for all types of animals, and then have more specific classes like "Dog" and "Cat" that inherit from the Animal class and add their own unique properties and methods.

2. What is the keyword used to declare a class?

```

public class Myclass {

}

```

3. In the class **Barang** at the **Part 2**, how many attributes owned by that class? What are they?

There are 4 attributes such as `namaBarang` with String, `jenisBarang` with String, `stok` with int, `hargaSatuan` with int.

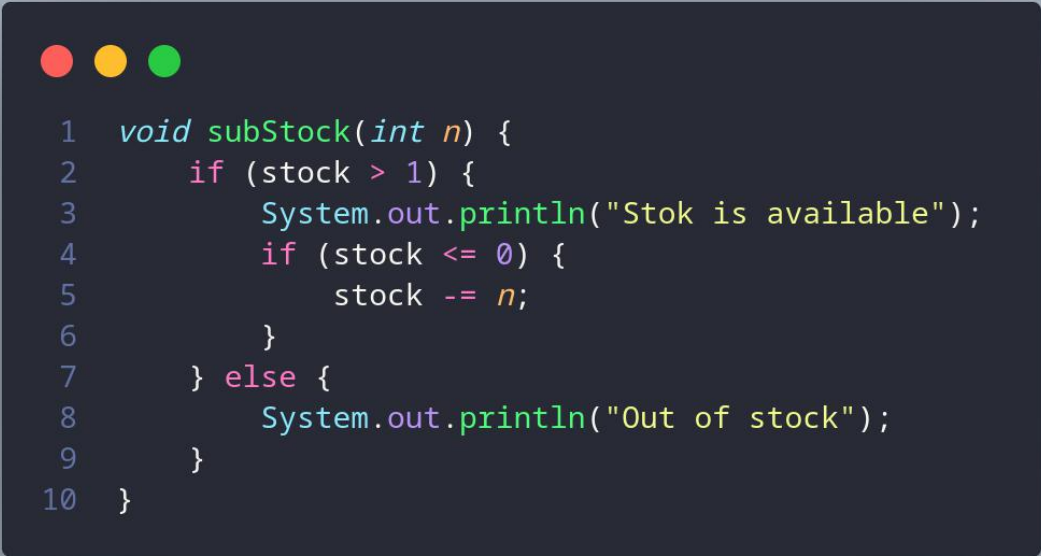
4. In the class **Barang** at the **Part 2**, on which line of code are the attributes declared?

line 4 and line 5

5. In the class **Barang** at the **Part 2**, how many methods owned by that class? What are they?

There are 4 method such as tampilBarang, tambahStok, kurangStok, hitungHargaTotal.

6. In the class **Barang** at the **Part 2**, on which line of code are the methods declared?
From line 7 to 21.
7. In the method **kurangiStok()** in class **Barang**, modify the method so that it will check the availability of **stok** before subtracting the **stok**! Then it will not be any subtraction if the **stok** is already less then or equals to zero.



```
1 void subStock(int n) {
2     if (stock > 1) {
3         System.out.println("Stok is available");
4         if (stock <= 0) {
5             stock -= n;
6         }
7     } else {
8         System.out.println("Out of stock");
9     }
10 }
```

8. Please give your explanation, why is method **tambahStok()** created with an int parameter? What is the use of that parameter int that method?
Because int parameter used to process number from outside method.
9. Why does method **hitungHargaTotal()** have a nonDvoid (int) data type? What is it for?
Because in method **hitungHargaTotal()** there is result which is return value of calculate price total.
10. Why does method **tambahStok()** have void data type?
The reason that the method **addStock()** has a void return type is because it does not need to return any value to the calling code.
11. In class **BarangMain**, in **Part 3**, on which line of code does the instantiation process run? And what is the name of the resulting object?
In line 5 an name object is b1.
12. How do you access the attributes and methods of the object?

nameObject.nameMethod();

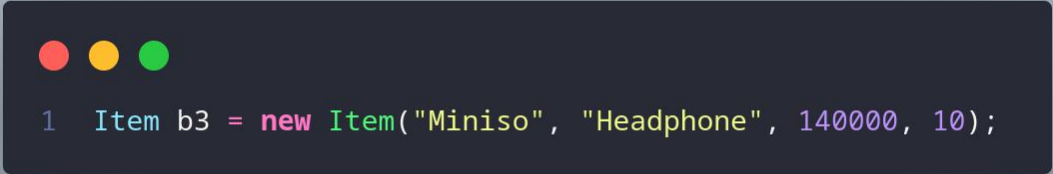
13. In class **Barang** in **Part 4**, on which line of code is the parametric constructor declared?

In line 9 to 14.

14. In class **BarangMain** in **Part 4**, what does actually we do on line of code 16?

To print attribute inside class.

15. Try to create another object called **b3** from class **barang** by using the parametric constructor of class **Barang**.



```
1 Item b3 = new Item("Miniso", "Headphone", 140000, 10);
```

12. TASK

1. Create the program based on the class diagram below!

Lingkaran
PHI: double r: double
hitungLuas(): double hitungKeliling(): double

Note:

Method hitungLuas() will calculate the area of the circle

Method hitungKeliling() will calculate the surrounding of the circle

Code:

```
1 public class Circle {
2     double phi = 3.14, r;
3     Circle() {
4
5     }
6     Circle(double r) {
7         this.r = r;
8     }
9     double calculateArea(double r) {
10        return phi * r * r;
11    }
12    double calculateCircumference(double r) {
13        return 2 * phi * r;
14    }
15 }
16
```

```

1 public class App {
2     public static void main(String[] args) throws Exception {
3         Circle circle1 = new Circle();
4         System.out.println(circle1.calculateCircumference(5));
5         System.out.println(circle1.calulateArea(5));
6     }
7 }
8

```

Result:

```

(zharsuke@asus-vivobook) - [~/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding]
$ /usr/bin/env /usr/lib/jvm/java-17-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/zharsuke/.config/Code/User/workspaceStorage/d617adc888ae15ed5f07d15a11b961e8/redhat.java/jdt_ws/coding_4679443b/bin App
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
31.400000000000002
78.5
(zharsuke@asus-vivobook) - [~/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding]
$

```

2. In the video game rental and shop, the most important data that they manage is RentalTransaction. It contains memberId, memberName, gameName, dailyPrice and dayRent (how many days it will be rent). It has a method to print the rental data and the price that should be paid by member. Please create a class diagram of the class and make the code!

Code:

```

1 public class RentalData {
2     int memberId;
3     String memberName, gameName;
4     double dailyPrice, day;
5     void printData() {
6         System.out.println("member Id = " + memberId);
7         System.out.println("member name = " + memberName);
8         System.out.println("Game Name = " + gameName);
9         System.out.println("Daily = " + dailyPrice);
10        System.out.println("Day = " + day);
11    }
12    double totalPrice(double dailyPrice, double day) {
13        return dailyPrice * day;
14    }
15 }

```

```

1 public class App {
2     public static void main(String[] args) throws Exception {
3         RentalData customer1 = new RentalData();
4         customer1.memberId = 1;
5         customer1.memberName = "Azhar";
6         customer1.gameName = "Resident Evil Village";
7         customer1.dailyPrice = 20_000;
8         customer1.day = 5;
9         customer1.printData();
10        System.out.println("Total Price = " + customer1.totalPrice(customer1.dailyPrice, customer1.day));
11    }
12 }
13

```

Result:

```

(zharsuke@asus-vivobook) - [~/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding]
$ cd /home/zharsuke/Documents/College/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding ; /usr/bin/env /usr/lib/
jvm/java-17-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/zharsuke/.config/Code/User/workspaceStorage/
d617adc888ae15ed5f07d15a11b961e8/redhat.java/jdt_ws/coding_4679443b/bin App
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
member Id = 1
member name = Azhar
Game Name = Resident Evil Village
Daily = 20000.0
Day = 5.0
Total Price = 100000.0

(zharsuke@asus-vivobook) - [~/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding]
$

```

3. Implement the code of this class diagram!

Item
name: String
unitPrice: int
qty: int
calculateTotalPrice(): int calculateDiscount(): int calculateFinalPrice(): int

Method calculateTotalPrice() will multiply the quantity of item and the unit price

Method calculateDiscount () will calculate the discount, with the role:

- o If total price > 100000, the discount will be 10%
- o If the total price between 50000 D 100000 the discount will be 5%
- o If the total price < 50000 it will be no discount

Method calculateFinalPrice () will calculate the price should be paid (total price minus discount)

Code:


```

1 public class App {
2     public static void main(String[] args) throws Exception {
3         Item keyboard = new Item();
4         keyboard.name = "Red Dragon";
5         keyboard.unitPrice = 350_000;
6         keyboard.qty = 2;
7         System.out.println("Total price = " + keyboard.calculateTotalPrice(keyboard.unitPrice, keyboard.qty));
8         System.out.println("Discount = " + keyboard.calculateDiscount());
9         System.out.println("Final price = " + keyboard.calculateFinalPrice());
10    }
11 }

```

```

1 public class Item {
2     String name;
3     int unitPrice, qty;
4     Item() {
5
6     }
7     Item(String name, int unitPrice, int qty) {
8         this.name = name;
9         this.unitPrice = unitPrice;
10        this.qty = qty;
11    }
12    int calculateTotalPrice(int unitPrice, int qty) {
13        return qty * unitPrice;
14    }
15    int calculateDiscount() {
16        int totalPrice = calculateTotalPrice(unitPrice, qty);
17        if (totalPrice > 100000) {
18            double totaltoDouble = (double) totalPrice;
19            double discounttoDouble = totaltoDouble * 0.10;
20            int discounttoInt = (int) discounttoDouble;
21            return discounttoInt;
22        } else if (calculateTotalPrice(unitPrice, qty) >= 50000 && calculateTotalPrice(unitPrice, qty) <= 100000) {
23            double totaltoDouble = (double) totalPrice;
24            double discounttoDouble = totaltoDouble * 0.05;
25            int discounttoInt = (int) discounttoDouble;
26            return discounttoInt;
27        } else {
28            return 0;
29        }
30    }
31    int calculateFinalPrice() {
32        return calculateTotalPrice(unitPrice, qty) - calculateDiscount();
33    }
34 }
35

```

Result:

```

(zharsuke@asus-vivobook) - [~/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding]
$ cd /home/zharsuke/Documents/College/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding ; /usr/bin/env /usr/lib/
jvm/java-17-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/zharsuke/.config/Code/User/workspaceStorage/
d617adc888ae15ed5f07d15a11b961e8/redhat.java/jdt_ws/coding_4679443b/bin App
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Total price = 700000
Discount = 70000
Final price = 630000

(zharsuke@asus-vivobook) - [~/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding]
$

```

4. Implement the code of class diagram below!

PacMa n
x: int y: int width: int height: int
moveLeft(): void moveRight(): void moveUp(): void moveDown(): void

printPosition(): void

Attribute x depicts the horizontal position/coordinate of Pacman, while attribute y depicts the vertical coordinate

Attribute width is for canvas width, and attribute height is for the height of the canvas

Method moveLeft() will move Pacman to the left (coordinate x will decrease), while moveRight() will move Pacman to the right (coordinate x will increase). The value of x will range from 0 to width value

Method moveUp() will move Pacman to the upper position (coordinate y will decrease), while moveDown() will move Pacman to the lower position (coordinate y will increase). The value of y must be between 0 to height value

Code:

```
1 public class App {
2     public static void main(String[] args) throws Exception {
3         PacMan pacMan = new PacMan();
4         pacMan.height = 500;
5         pacMan.width = 400;
6         pacMan.x = 0;
7         pacMan.y = 0;
8         System.out.println("Before move");
9         pacMan.printPosition();
10        System.out.println();
11        System.out.println("After move");
12        pacMan.moveRight();
13        pacMan.printPosition();
14    }
15 }
16
```

```

1  public class PacMan {
2      int x, y, width, height;
3      PacMan() {
4
5      }
6      PacMan(int x, int y, int width, int height) {
7          this.x = x;
8          this.y = y;
9          this.width = width;
10         this.height = height;
11     }
12     void moveLeft() {
13         if (x >= 0 && x <= width) {
14             x--;
15         } else {
16             System.out.println("cannot move!");
17         }
18     }
19     void moveRight() {
20         if (x >= 0 && x <= width) {
21             x++;
22         } else {
23             System.out.println("cannot move!");
24         }
25     }
26     void moveUp() {
27         if (y >= 0 && y <= height) {
28             y--;
29         } else {
30             System.out.println("cannot move!");
31         }
32     }
33     void moveDown() {
34         if (y >= 0 && y <= height) {
35             y++;
36         } else {
37             System.out.println("cannot move!");
38         }
39     }
40     void printPosition() {
41         System.out.println("Height = " + height);
42         System.out.println("width = " + width);
43         System.out.println("Coordinate pacman = (" + x + ", " + y + ")");
44     }
45 }
46

```

Result:

```

(zharsuke@asus-vivobook) - [~/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding]
$ cd /home/zharsuke/Documents/College/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding ; /usr/bin/env /usr/lib/
jvm/java-17-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/zharsuke/.config/Code/User/workspaceStorage/
d617adc888ae15ed5f07d15a11b961e8/redhat.java/jdt_ws/coding_4679443b/bin App
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Before move
Height = 500
width = 400
Coordinate pacman = (0,0)

After move
Height = 500
width = 400
Coordinate pacman = (1,0)

(zharsuke@asus-vivobook) - [~/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_2/coding]
$

```