清森学校
BEIJING QINGSEN SCHOOL

AP-CSA
Using Objects

YING HUANG

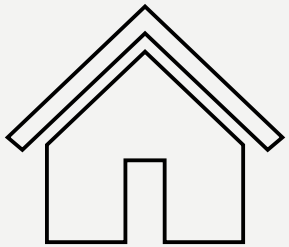
SEP.2022

CLASS & OBJECT

清森学校
BEIJING QINGSEN SCHOOL

THE RELATIONSHIP CLASSES AND OBJECTS

Class: house



Attribute:
Color
Owner
…

Behavior:
ChangeColor()

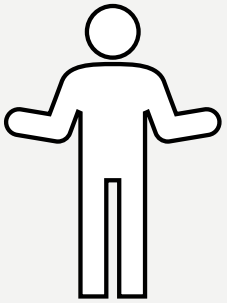
Ying

Jerry

Stan

THE RELATIONSHIP CLASSES AND OBJECTS

Class: Person



Attribute: ??

Behavior:??

CLASS & OBJECT

A class is a blueprint for creating objects with the same behavior and defined attributes.

An object is a specific entity, made from a class, that you can manipulate in your programs.

Objects are instances of classes with variables used to name them

Give a example about object and class in the really world.

清森学校
BEIJING QINGSEN SCHOOL

CLASS & OBJECT

Following with coding to create a class house !

Step 1. Create a java file, naming house

Step 2. declare the house class's attributes(like color, owner, ID…)

Step 3. **Constructor(Method)**

CONSTRUCTOR 构造函数

The constructor of a class is a method that allows us to initialize the attributes(variables) of an object when it is first created.

The name of constructor is same as class's name!

Syntax:

```
public className(…)
{
    ….
}
```

OVERLOADED CONSTRUCTORS

Constructors are said to be **overloaded** when there are multiple constructors with the same name but a different signature.

```
public house()
{
    ....
}
```

no parameter

```
public house(String color)
{
    ....
}
```

one parameter

A parameter is a variable used to define a particular value during a function definition.

OVERLOADED CONSTRUCTORS

We can call different constructors to initialize our objects.

```
house h1 = new house();
// default constructor initializes
house h2 = new house("Red");
// h2.color = red
house h3 = new house("Red","Ying",111);
// h3.color = Red, h3.owner  =Ying h3.ID =111
```

CREATE A OBJECT

An object variable is created using the keyword **new** followed by a call to a constructor.

Syntax:

```
className variableName = new className(…);

//example
house h1 =new house();
house h2 = new house("Green", "Ying"..)
```

USING A OBJECT

We can access the attributes of an object by using the **dot notation**.

Syntax:

```
variableName.attribute  =  value;

//example
h1.color  =  "Red";
h1.ID  =  "111";
```

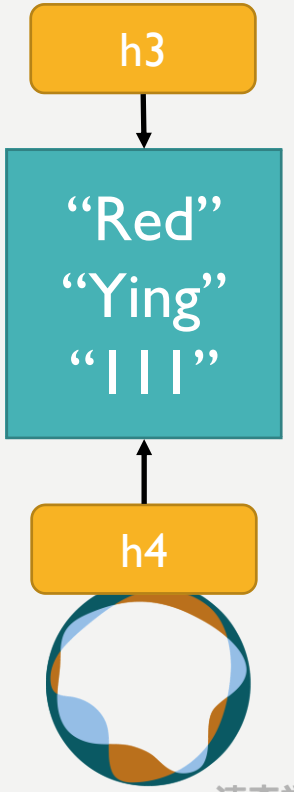
PRINT OBJECT

```
System.out.print(object); // the address of object
```

PRIMITIVE VS. REFERENCE TYPE

While the memory associated with a variable of a reference type holds an object reference value. This value is the memory address of the referenced object.

```
house h3 = new house("Red","Ying",111)
house h4 = h3;
// h3 and h4 stores the same address in memory therefore both refer to the same object.
```



PRIMITIVE VS. REFERENCE TYPE

The memory associated with a variable of a **primitive type(int, double, boolean)** holds an actual primitive value.

```
int num1 = 3;
// the memory associated with x actually holds the value 3
int num2 = num1;
// the value of num2 copies from num1, num2 has memory
to hold 3.
```

Here we have two different integers in different memory both of which has the value 3.

METHOD

清森学校
BEIJING QINGSEN SCHOOL

MODULARITY

- **Modularity:** Writing code in smaller, more manageable components or modules. Then combining the modules into a cohesive system.

  In modularity, break complex code into smaller tasks and organize it using methods.

Method define the behaviors or functions for objects.

A **Method** is a named group of programming instructions that accomplish a specific task.

EXAMPLE

Consider the following code which asks the user to enter two numbers and print out the average.

```
Scanner console = new Scanner(System.in);
System.out.print("Enter a number: ");
int num1 = console.nextInt();
System.out.print("Enter a number: ");
int num2 = console.nextInt();
System.out.println("The average is " + (num1 + num2)/2.0);
```

METHOD 方法

A **method** is a named group of programming instructions that accomplish a specific task.

Example:

```
public double printArea()
{…}
```

1. **Access specifier/modifiers:** public
2. **Return type:** double
3. **Method name:** printArea
4. **Parameter list:** none
5. **Method body:**{…}

METHOD SIGNATURE

The **method signature** is the combination of the method name and the parameter list.

Example:

```
printArea()
```

PARAMETERS 参数

- A **parameter** is a variable used to define a particular value during a function definition.
- The parameters in the method header are **formal parameters**.

```
public void printArea(int width, int height)
{ int are =width*height; }
```

- The parameters in the method signature are **actual parameters/ arguments**.

```
printArea(5, 10)
```

RETURN TYPE

- **return:** To send out a value as the result of a method.
- The opposite of a parameter:
  - *Parameters* send information *in* from the caller to the method.
  - *Return* values send information *out* from a method to its caller.
- **Returned values** can be stored in a variable, used in other math expressions or printed on the console.

```
public double printArea()
{…}
```

NO-RETURN- VOID

- Void methods do not have return values.

- Void methods do not have return values and are therefore not called as part of an expression.

```
public void printArea()
{…}
```

EXERCISE

1. Create two java files: Rectangle.java and RectangleTester.java
2. Rectangle.java will include width and height as their attributes.
3. Rectangle.java will have a method to change the width of object. (no static method)
4. Rectangle.java will have a method to calculate the area of Rectangle and return the area.

CODEHS.COM

Each student should signup at

https://codehs.com/go/2F8C6

Finish

2.4.5-2.4.7

2.5.5-2.5.8

2.6.6-2.6.8

清森学校
BEIJING QINGSEN SCHOOL

SCREENSHOT

Practice | Resume | Ying Huang

2.4.5 Hello!

Submit + Continue | Save

HelloTester.java

```java
import java.util.Scanner;

public class HelloTester
{
    public static void main(String[] args)
    {
        // Create a Scanner object
        Scanner input = new Scanner(System.in);

        System.out.println("Please enter your name: ");
        String name = input.nextLine();

        Hello greeting = new Hello(name);

        //Answers may vary slightly here
        greeting.english();
        greeting.russian();
        greeting.french();

    }
}
```

Output | Test Cases | Docs | Assignment | Grade | More

Test Cases

Check Code | Minimize | Expand

2/2

	Pass	Test	Message
>	✓	You should create one Hello object	Great!
>	✓	You should print three greetings	Great!

清森学校
BEIJING QINGSEN SCHOOL