COSC 111:
Computer Programming I

Dr. Bowen Hui

University of British Columbia
Okanagan

Review

- Class template:

```
class ClassName
{
        // attributes
        // methods
}
```

- Method template:

```
returnType methodName( varType1 var1, …, varTypek vark )
{
        // commands go here
        return value;        // omitted if returnType is void
}
```

- Use this template to create methods that go inside classes

Review (cont.)

- TestClass template:

```
class TestClassName
{
    public static void main( String[] args )
    {
        // create objects
        // call objects' methods
    }
}
```

- This template is actually the combination of the previous two!

Control Flow

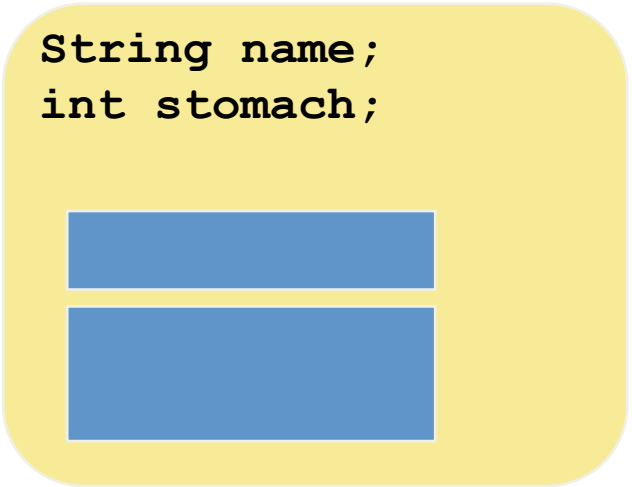
- Text p.47: “A class is used to create objects just as a house blueprint is used to create different, but similar, houses”
- EX: Dog class – attributes and behaviour of a dog
  - casper is a Dog object
  - bitzy is another Dog object
- When we run the program, we type:

  > java TestDog
- Run the commands inside main() method in top-down fashion
- Note about a TestClass:
  - No attributes
  - Only one method called main()

TestDog Example

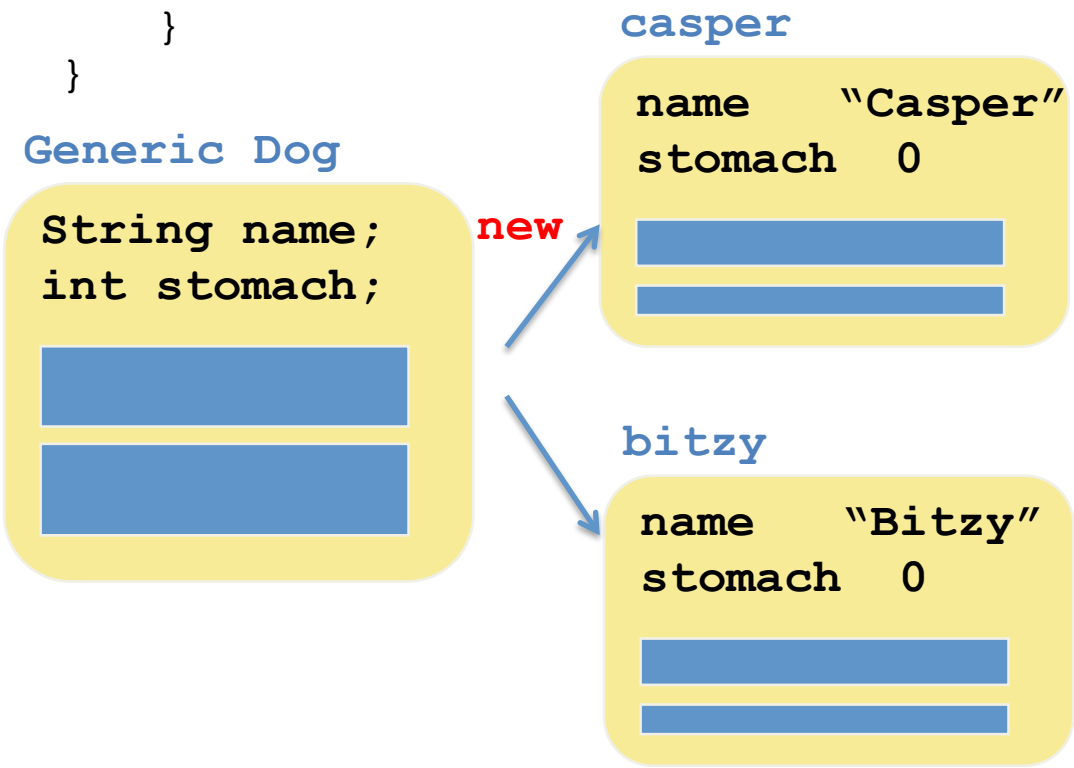
```
class TestDog
{
    public static void main( String[] args )
    {
        Dog casper = new Dog( "Casper" );
        Dog bitzy = new Dog( "Bitzy" );
        casper.eatSnacks( 2 );
        bitzy.eatSnacks( 6 );
    }
}
class Dog
{
    String name;
    int stomach;
    Dog( String n ) {…}
    eatSnacks( int num ) {…}
}
```

Generic Dog structure (from Dog class):



TestDog Example (cont.)

```
class TestDog
{
        public static void main( String[] args )
        {
                Dog casper = new Dog( "Casper" );
                Dog bitzy = new Dog( "Bitzy" );
                casper.eatSnacks( 2 );
                bitzy.eatSnacks( 6 );
        }
}
```

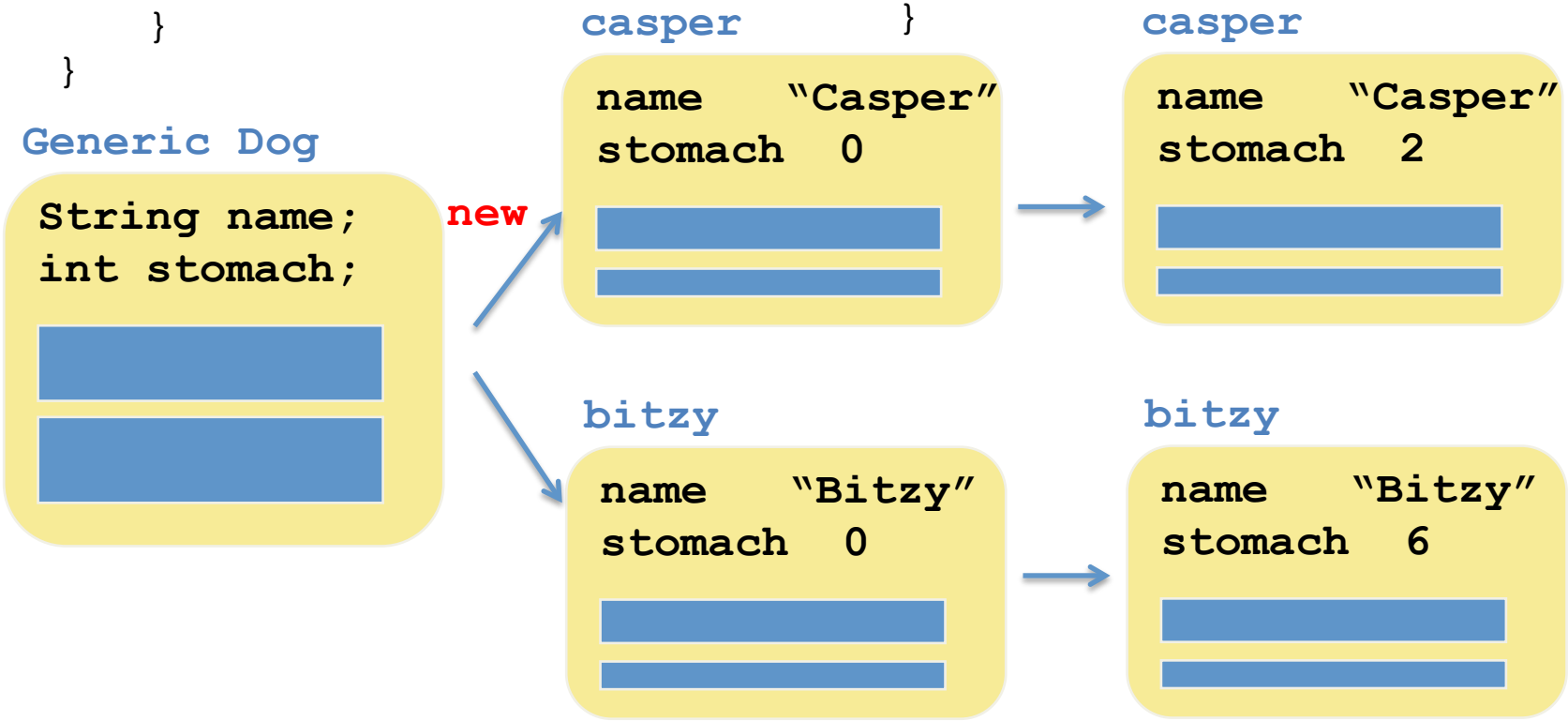



```
class Dog
{
        String name;
        int stomach;
        Dog( String n )
        {
                name = n;
                stomach = 0;
        }
        // other methods
}
```

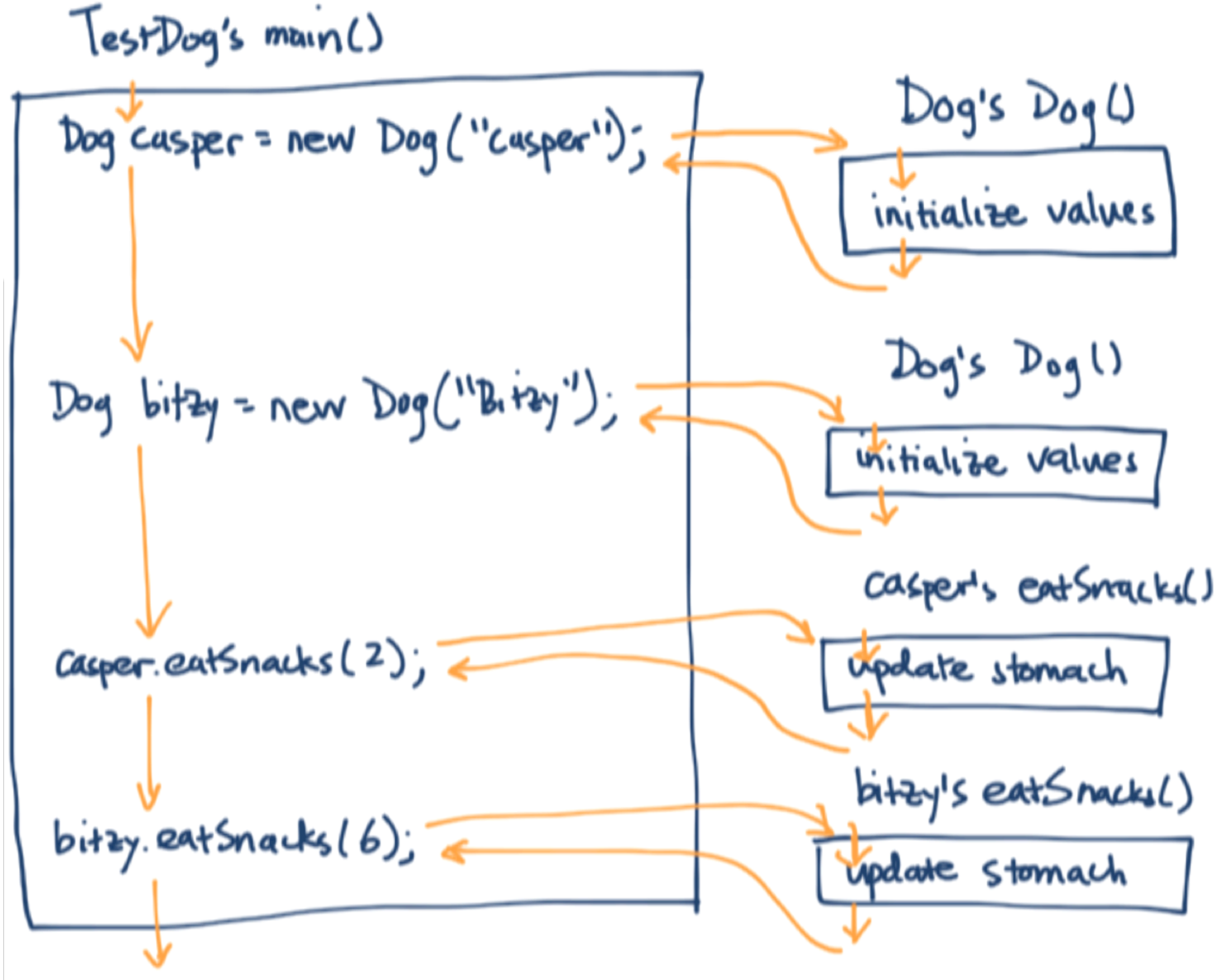
TestDog Example (cont.)

```
class TestDog
{
        public static void main( String[] args )
        {
                Dog casper = new Dog( "Casper" );
                Dog bitzy = new Dog( "Bitzy" );
                casper.eatSnacks( 2 );
                bitzy.eatSnacks( 6 );
        }
}
```

```
class Dog
{
        // …
        void eatSnacks( int num )
        {
                stomach = stomach + num;
        }
}
```



Method Control Flow

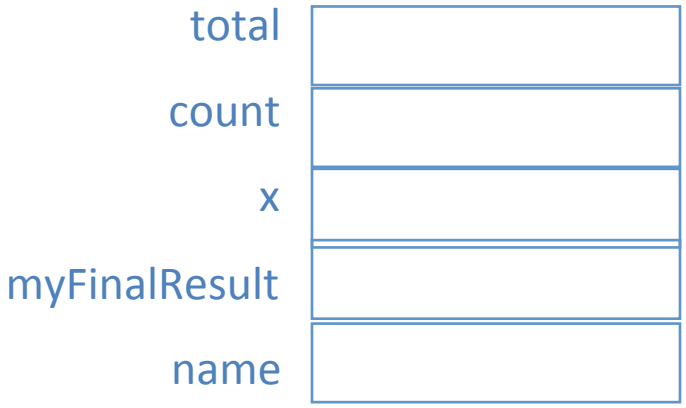


What goes inside methods?

- Commands to express how the process is to be carried out
- Written as Java statements
  - Use variables, methods
  - Use special capitalization and punctuation
  - Avoid reserved words – words that have special, pre-defined meaning in the Java language (see Ch1.4)

Variables

- A label for a location in memory that holds a value
- Variable declaration specifies:
  - Type of information
  - Label to refer to that variable with
- EX:
  - int total;
  - int count, x, myFinalResult;
  - String name;



- Multiple vars can be declared on the same line if they have the same data type
- No two vars can have the same name (for now)
- Use camel case for naming variables

Initialization and Assignment

- When an equal sign is used in a statement, a value is assigned to a variable
- The first time a variable gets a value is called an initialization
- No special name for subsequent assignments
- EX:

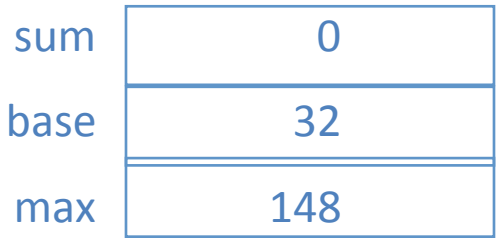
```
int sum = 0;
int base = 32;
int max = 148;
…
sum = sum + 1;
base = max / 2;
max = sum + base;
```

- When var is used in a statement, the current value stored in that var is used

Initialization and Assignment

- When an equal sign is used in a statement, a value is assigned to a variable
- The first time a variable gets a value is called an initialization
- No special name for subsequent assignments
- EX:

```
int sum = 0;
int base = 32;
int max = 148;
…
sum = sum + 1;
base = max / 2;
max = sum + base;
```

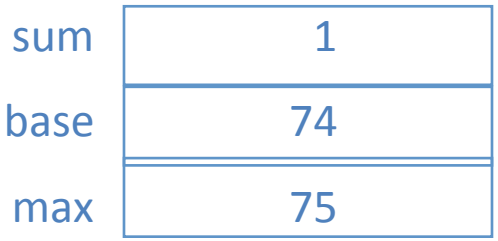


- When var is used in a statement, the current value stored in that var is used

Initialization and Assignment

- When an equal sign is used in a statement, a value (on the RHS) is assigned to a variable (on the LHS)
- The first time a variable gets a value is called an initialization
- No special name for subsequent assignments
- EX:

```
int sum = 0;
int base = 32;
int max = 148;
…
sum = sum + 1;
base = max / 2;
max = sum + base;
```



- When var is used in a statement, the current value stored in that var is used

Data Types

- We have seen:
  - int
  - String
  - boolean
  - Others created based on your own classes
- In reality, Java has 8 primitive types:
  - 4 for integers: byte, short, int, long
  - 2 for decimal numbers: float, double
  - 1 for characters: char
  - 1 for truth values: boolean
- See Ch2 for more details
- Note: String is not a primitive type!

Truth Values

- Use boolean to represent when something can only be true or false
- EX:
  - Is the result found?
  - Is the coin fake?
  - Is the gas tank full?
- Use reserved words: true, false
- Method example:

```
boolean isFull( int gasTank )
{
        // check number of litres left in tank
        return false;
}
```

Characters

- A char stores a single character used with single quotes
- EX:
  'a', 'X', '7', '$', ' ', '\n'
- Example declarations:
  char topGrade = 'A';
  char separator = ' ';
  char terminator = ';';
- See text for range of characters supported

Strings

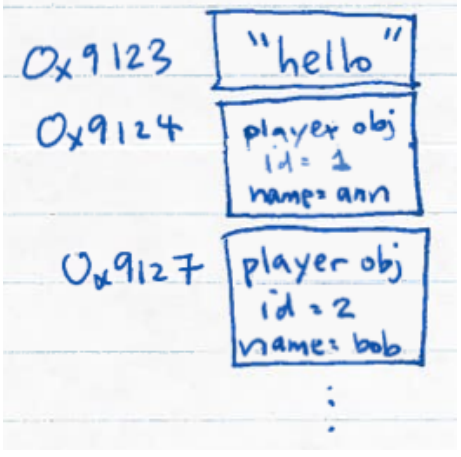
- Holds more than one character in a sequence inside double quotes
- EX:
  “this is a phrase”
  “3333 University Way, Kelowna B.C., V1V 1V7”
  “x”
  “27”
  “y = 5”
- The following are equivalent:
  String question1 = new String(“really?”);
  String question2 = “really?”;
- We use the latter shortcut for convenience
- More on String operations in Lab 3

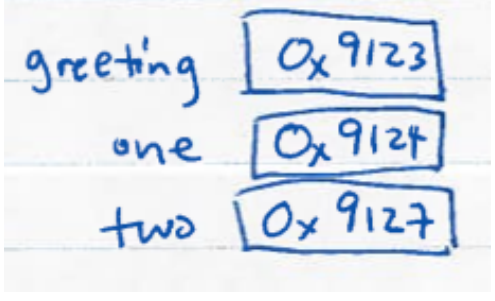
Object variables

- Labels to the memory location where objects are stored in memory (physical address)
- Use new to instantiate object variables
- EX:

```
String  greeting = new String("hello");
Player  one      = new Player(1, "ann");
Player  two      = new Player(2, "bob");
```

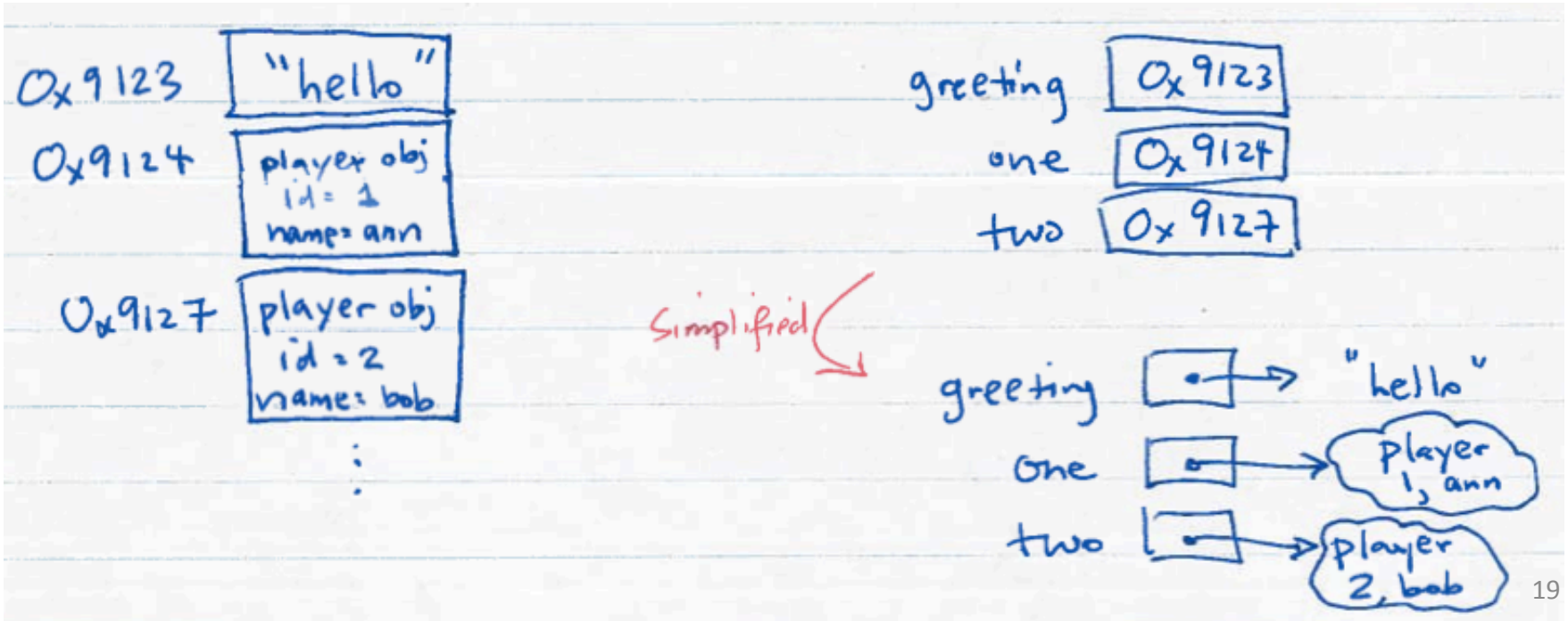
- In memory:



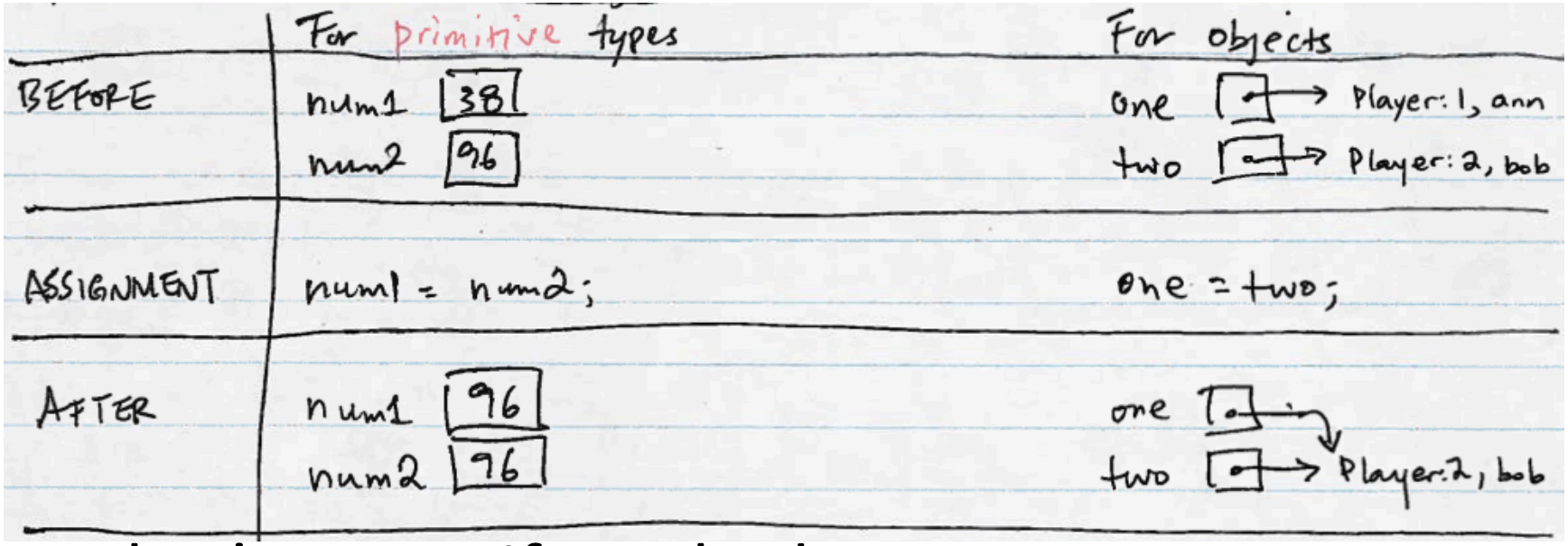


Object variables (Simplified)

- Now, an object variable is just a label that holds a pointer of the object
  - A pointer is a reference to a physical address
  - A null pointer is when the address has nothing



Different Assignment Behaviours



- What's more, if you had:

  two.name = "cam";

- Or, later, you also had:

  Player three = new Player(3, "dan");

  one = three;

Statements

- A way to issue a command in Java
- Like a sentence in English
- Ends in semicolon
- Can be made up of:
  - Variable declarations and assignments
  - Method calls
  - Arithmetic expressions
  - Control statements – special patterns (later)
- Can have a statement block by grouping multiple statements inside { … }
  - Where have we seen these before?

Arithmetic expressions

- Compute numeric results using arithmetic operators +, –, *, /, %
- EX:
  2 + 3
  12 / 3
- How is an operator different from a method?
- Operands = Input parameters for operators
- Where type matters:
  if one/both operands are decimal types, then the resulting value will also be a decimal

Examples

12 / 2 = ?

12.0 / 2 = ?

10 / 4.0 = ?

10 / 4 = ?

4.0 / 10 = ?

4 / 10 = ?

12 % 3 = ?

10 % 3 = ?

Examples

12 / 2 = 6

12.0 / 2 = ?

10 / 4.0 = ?

10 / 4 = ?

4.0 / 10 = ?

4 / 10 = ?

12 % 3 = ?

10 % 3 = ?

Examples

12 / 2 = 6

12.0 / 2 = 6.0

10 / 4.0 = ?

10 / 4 = ?

4.0 / 10 = ?

4 / 10 = ?

12 % 3 = ?

10 % 3 = ?

Examples

12 / 2 = 6

12.0 / 2 = 6.0

10 / 4.0 = 2.5

10 / 4 = ?

4.0 / 10 = ?

4 / 10 = ?

12 % 3 = ?

10 % 3 = ?

continue (check in Java)

Operator Precdence

- Operators can be combined like Math expressions
- EX: result = total + count / max – offset;
- Order of evaluation:
  ( )
  left to right
  *, / , %
  +, –
- EX: a / (b + c) + d % e
- Assignment operator = has lower precedence than arithmetic operators
- EX: answer = sum / 4 + max * lowest;
- The same var can appear on both sides
- EX: counter = counter + 1;

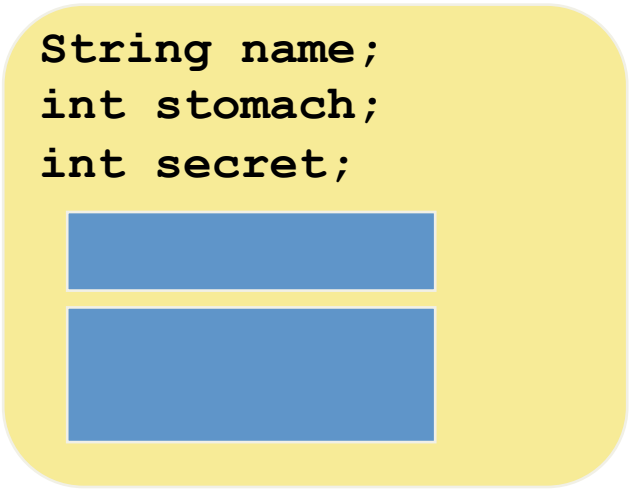
Shorthand Notation

operator	example	equivalent to
++	x++;	x = x + 1;
+=	x += y;	x = x + y;
-=	x -= y;	x = x - y;
*=	x *= y;	x = x * y;
/=	x /= y;	x = x / y;
%=	x %= y;	x = x % y;

Method calls for communication: TestDog v2 Example

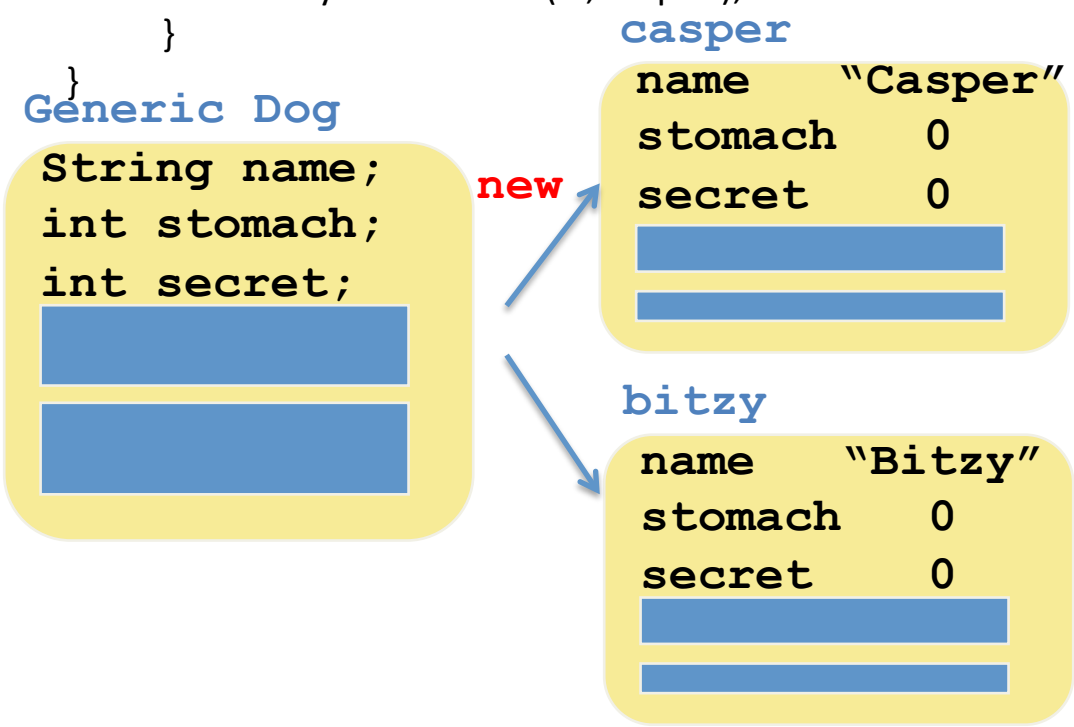
```
class TestDog
{
    public static void main( String[] args )
    {
        Dog casper = new Dog( “Casper” );
        Dog bitzy = new Dog( “Bitzy” );
        casper.eatSnacks( 2 );
        bitzy.eatSnacks( 6 );
        bitzy.shareSnacks( 1, capser );
    }
}
```

Generic Dog structure (from Dog class):



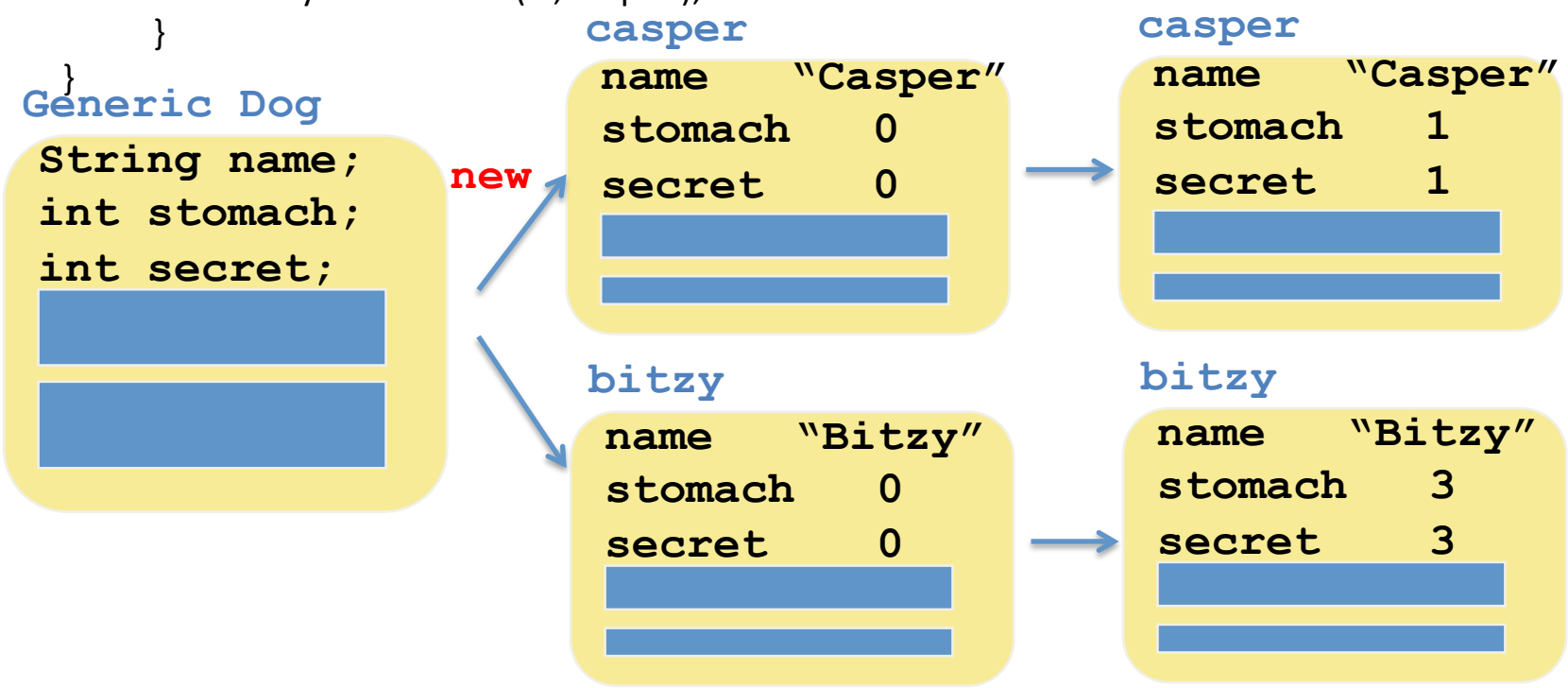
TestDog Example (cont.)

```
class TestDog
{
        public static void main( String[] args )
        {
                Dog casper = new Dog( "Casper" );
                Dog bitzy = new Dog( "Bitzy" );
                casper.eatSnacks( 2 );
                bitzy.eatSnacks( 6 );
                bitzy.shareSnacks( 1, casper );
        }
}
```



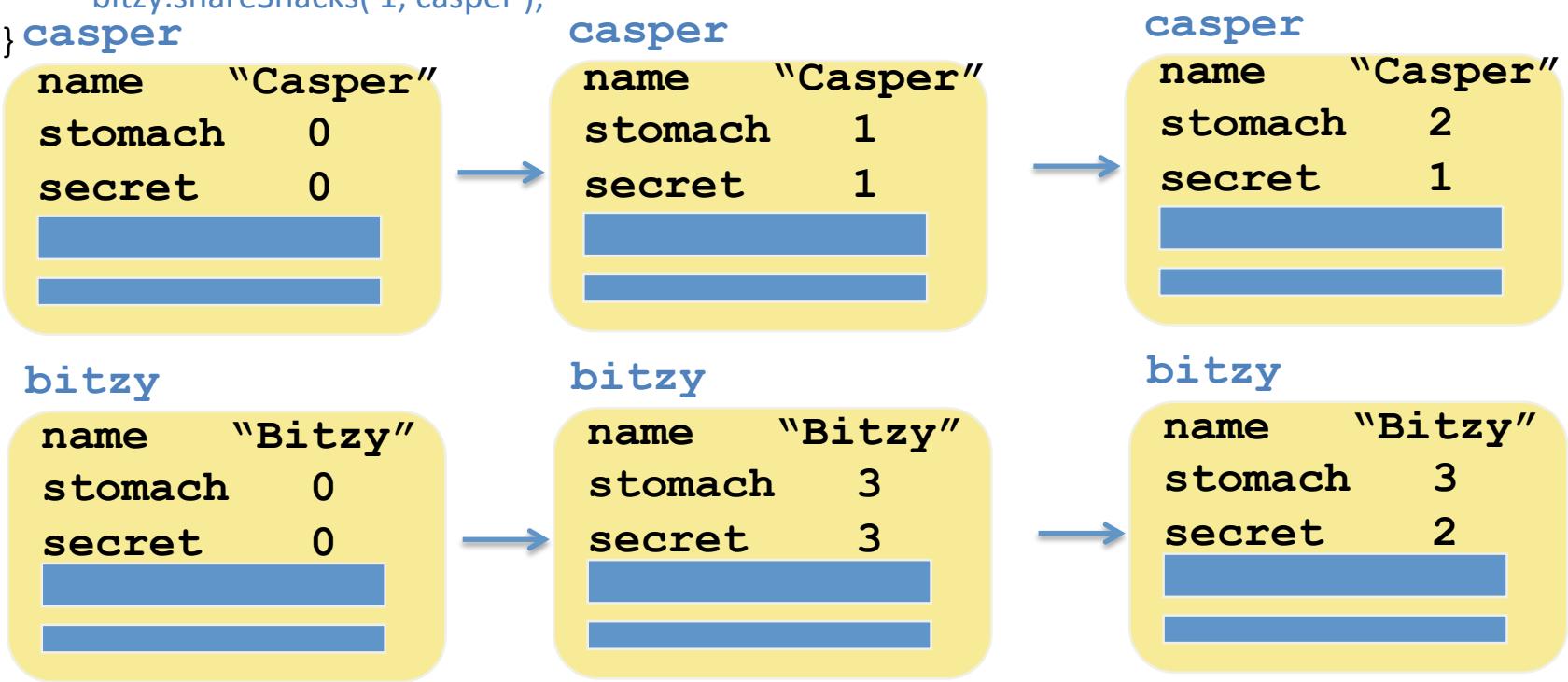
TestDog Example (cont.)

```
class TestDog
{
        public static void main( String[] args )
        {
                Dog casper = new Dog( "Casper" );
                Dog bitzy = new Dog( "Bitzy" );
                casper.eatSnacks( 2 );
                bitzy.eatSnacks( 6 );
                bitzy.shareSnacks( 1, casper );
        }
}
```



TestDog Example (cont.)

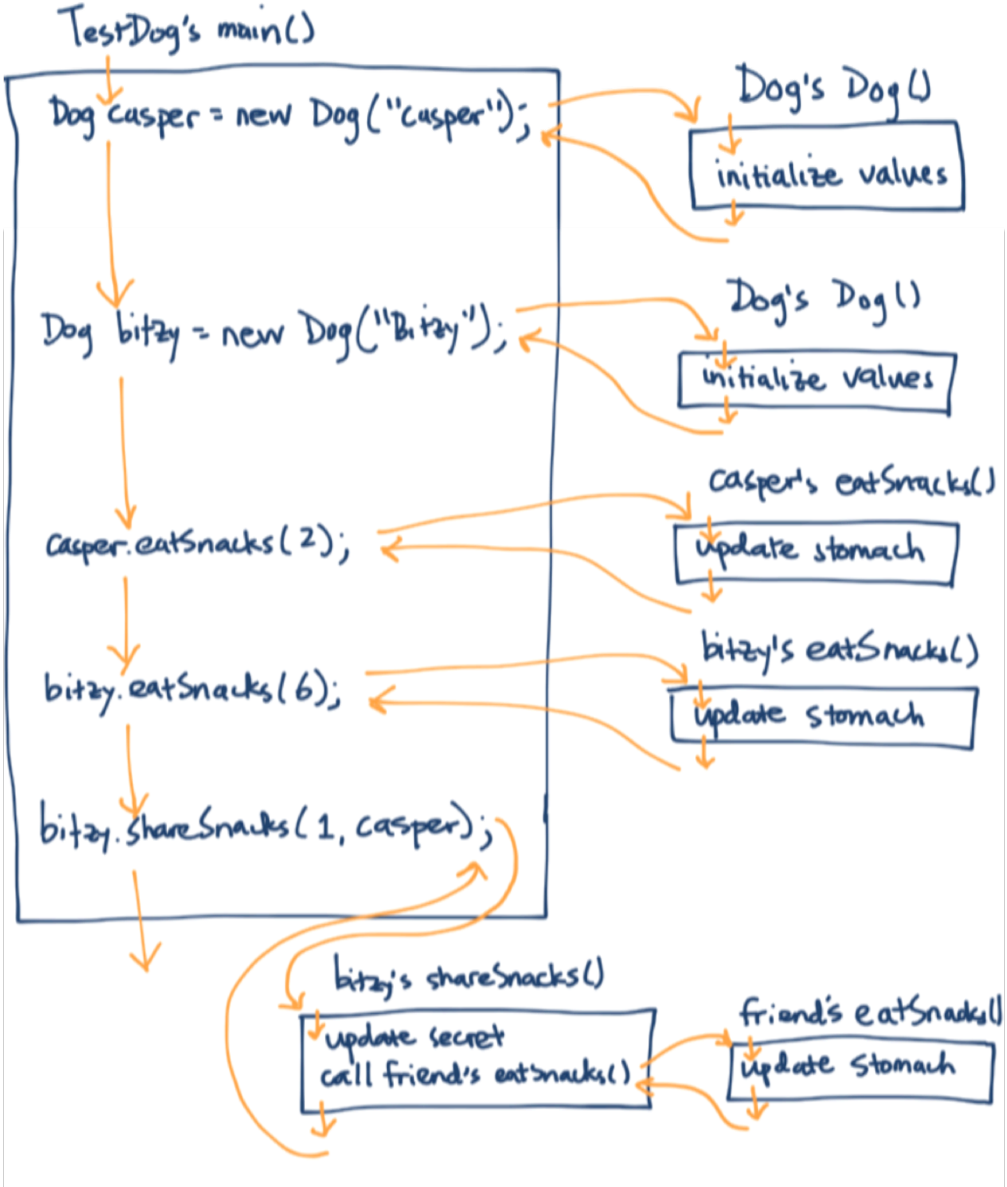
```
class TestDog
{
        public static void main( String[] args )
        {
                Dog casper = new Dog( "Casper" );
                Dog bitzy = new Dog( "Bitzy" );
                casper.eatSnacks( 2 );
                bitzy.eatSnacks( 6 );
                bitzy.shareSnacks( 1, casper );
        }
}
```



The shareSnacks() method

- Given:
  bitzy.shareSnacks(1, casper);
- Change eatSnacks() method to:
  - Divide numCookies by 2
  - Add half of numCookies to stomach attribute (eat them)
  - Add other half to secret attribute (safe them up)
- Add a new method inside Dog class
  void shareSnacks(int numCookies, Dog friend) { … }
- Inside shareSnacks():
  - Subtract numCookies from secret
  - Call friend's eatSnacks() method with numCookies

Method Control Flow



Class Interactions in A1

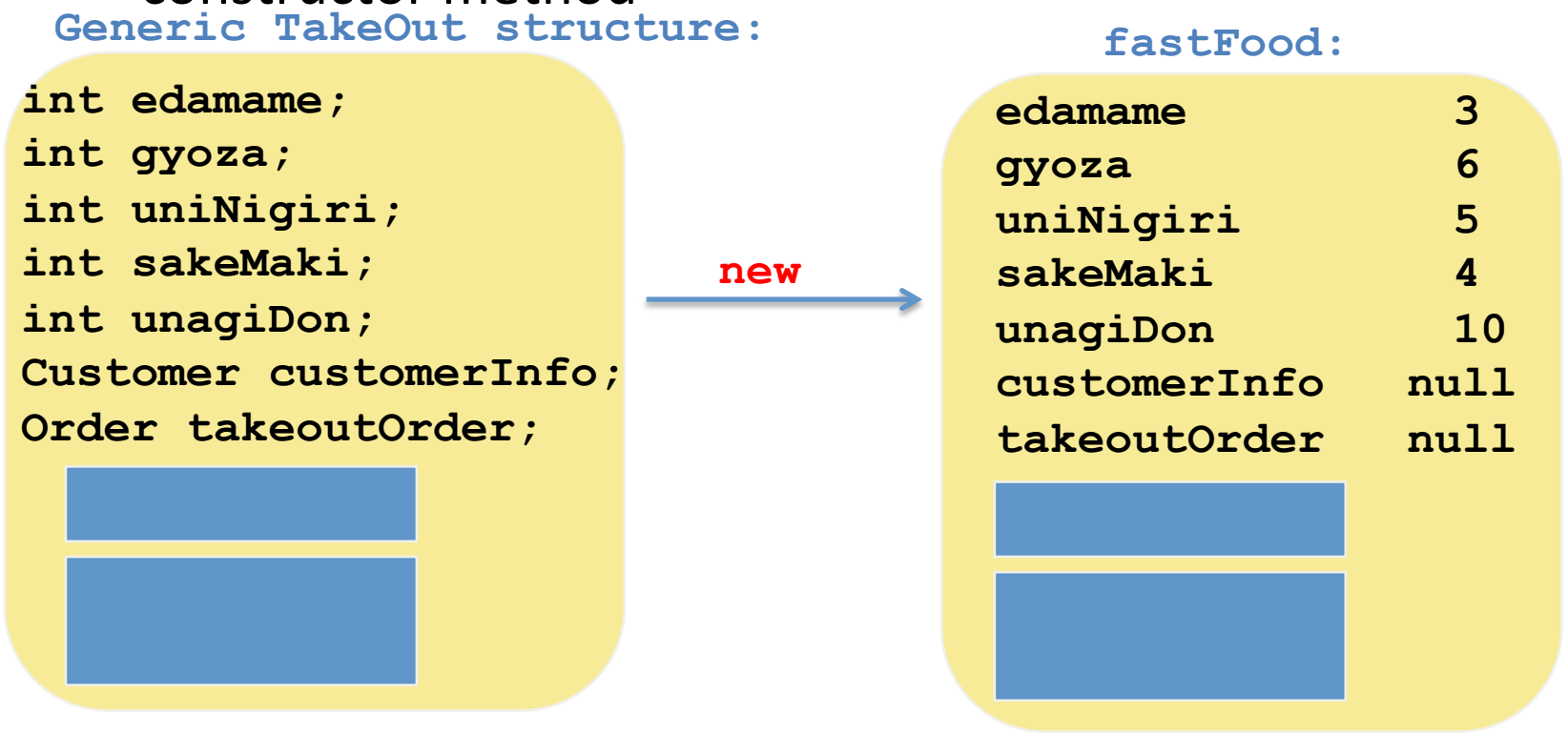
```
class TakeOut
{
   // attributes
   int edamame;
   int gyoza;
   int uniNigiri;
   int sakeMaki;
   int unagiDon;
   Customer customerInfo;
   Order takeoutOrder;

   // methods
   // …
}
```

Generic TakeOut structure:

```
int edamame;
int gyoza;
int uniNigiri;
int sakeMaki;
int unagiDon;
Customer customerInfo;
Order takeoutOrder;
```

Creating Objects

- Happens when we create new objects by calling the class constructor method



- Where is the TakeOut constructor method called?

Continuing with the next statement in the TestTakeOut class …

- What happens when we call:
  fastFood.makeNewCustomer(“Jen”, 1234567);
  - In which class is this method defined?
  - What do the comments there say this method needs to do?
- The method header is:
  void makeNewCustomer(String name, int phone) { … }
  - What is the value substituted into the variable name?
  - What is the value substituted into the variable phone?
  - How do we create a new Customer object in this method?
  - How do we create a new Order object in this method?

From Customer.java

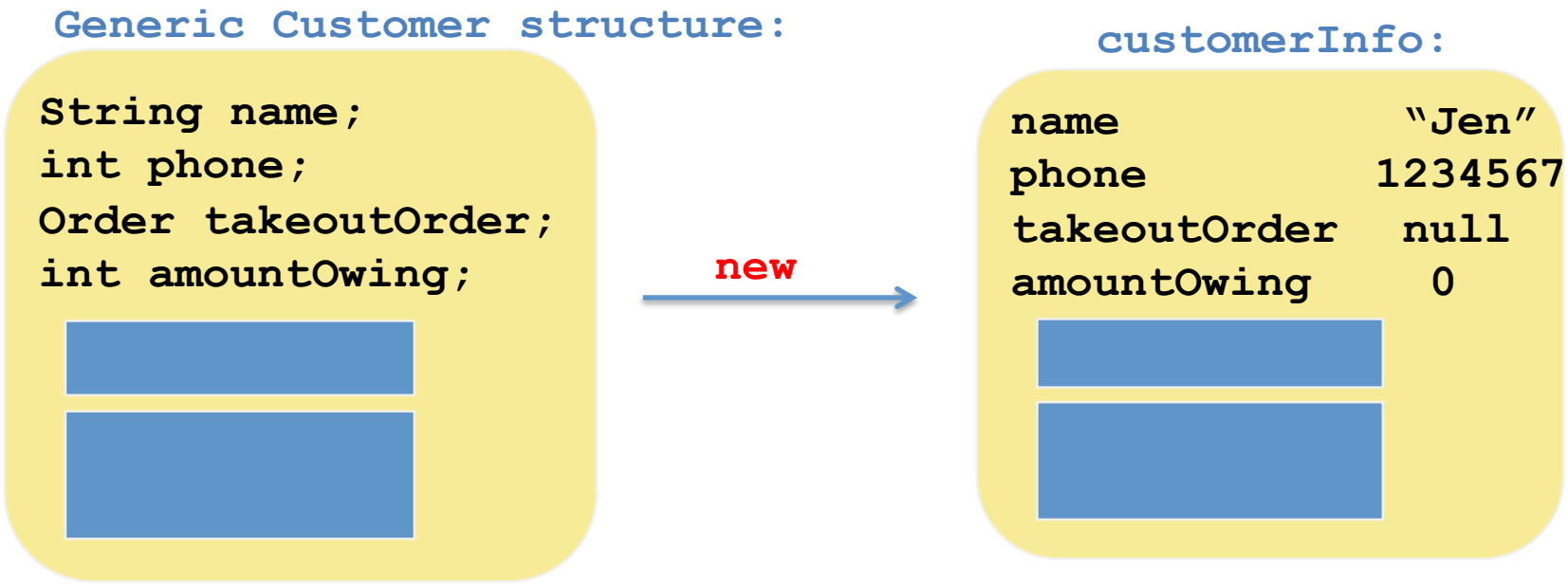
```
class Customer
{
   // attributes
   String name;
   int phone;
   Order takeoutOrder;
   int amountOwing;

   // methods
   // …
}
```

Generic Customer structure:

```
String name;
int phone;
Order takeoutOrder;
int amountOwing;
```

Creating Objects

- Happens when we create new objects by calling the class constructor method



- Copy the class definition to get same structure
- Assign constructor parameter values to attributes if available
- Assign default values to remaining attributes
  (statements inside constructor method will tell you how)

In makeNewCustomer()

- The missing code must create a new Customer object as well as a new Order object

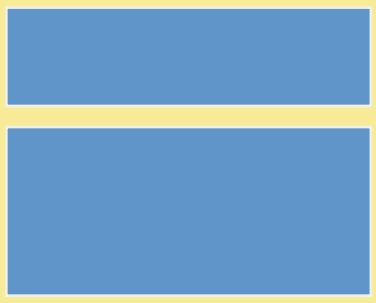
Generic Order structure:

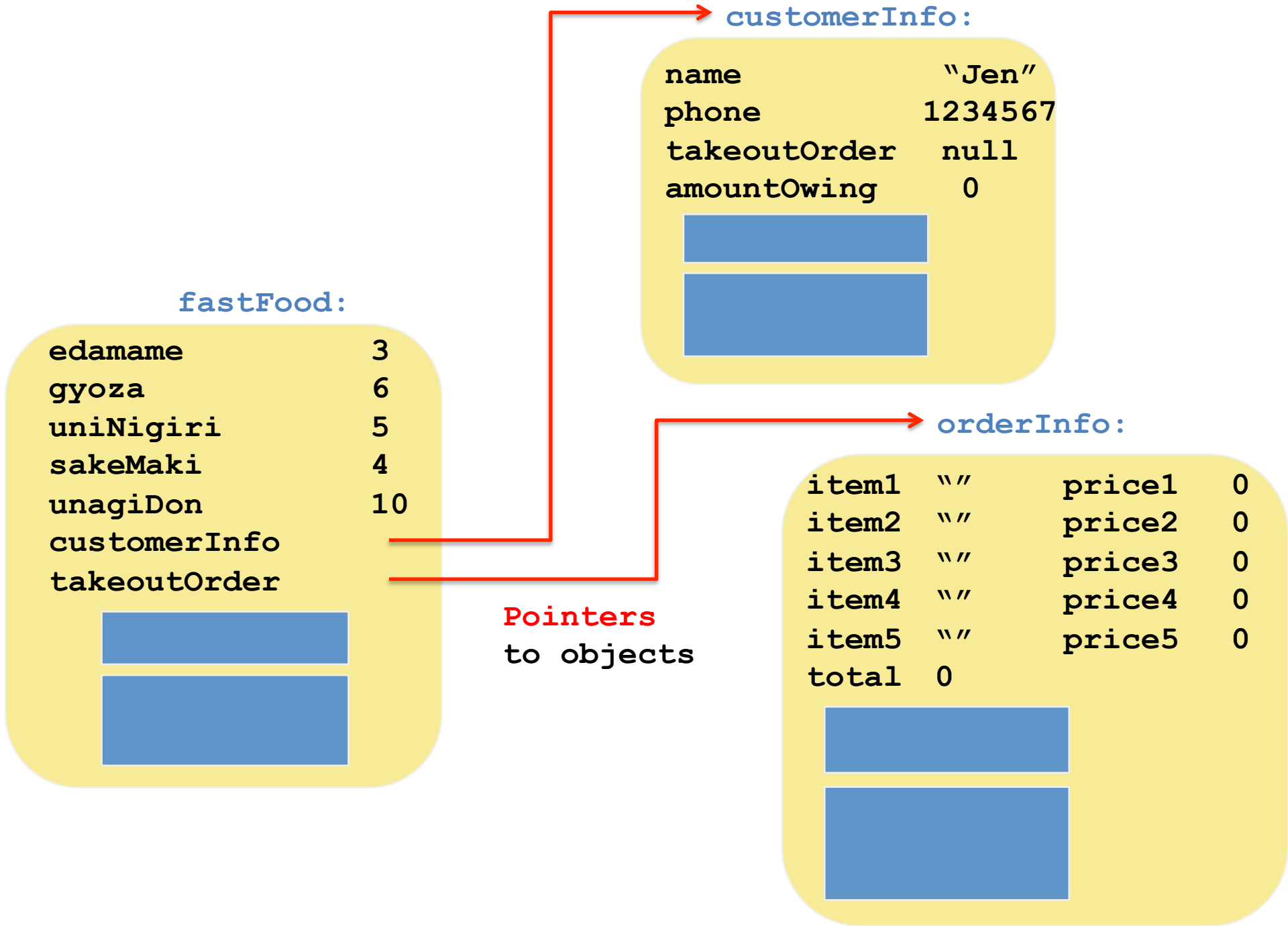
```
int item1, item2,
item3, item4, item5;
int price1, price2,
price3, price4,
price5;
int total;
```

new

orderInfo:

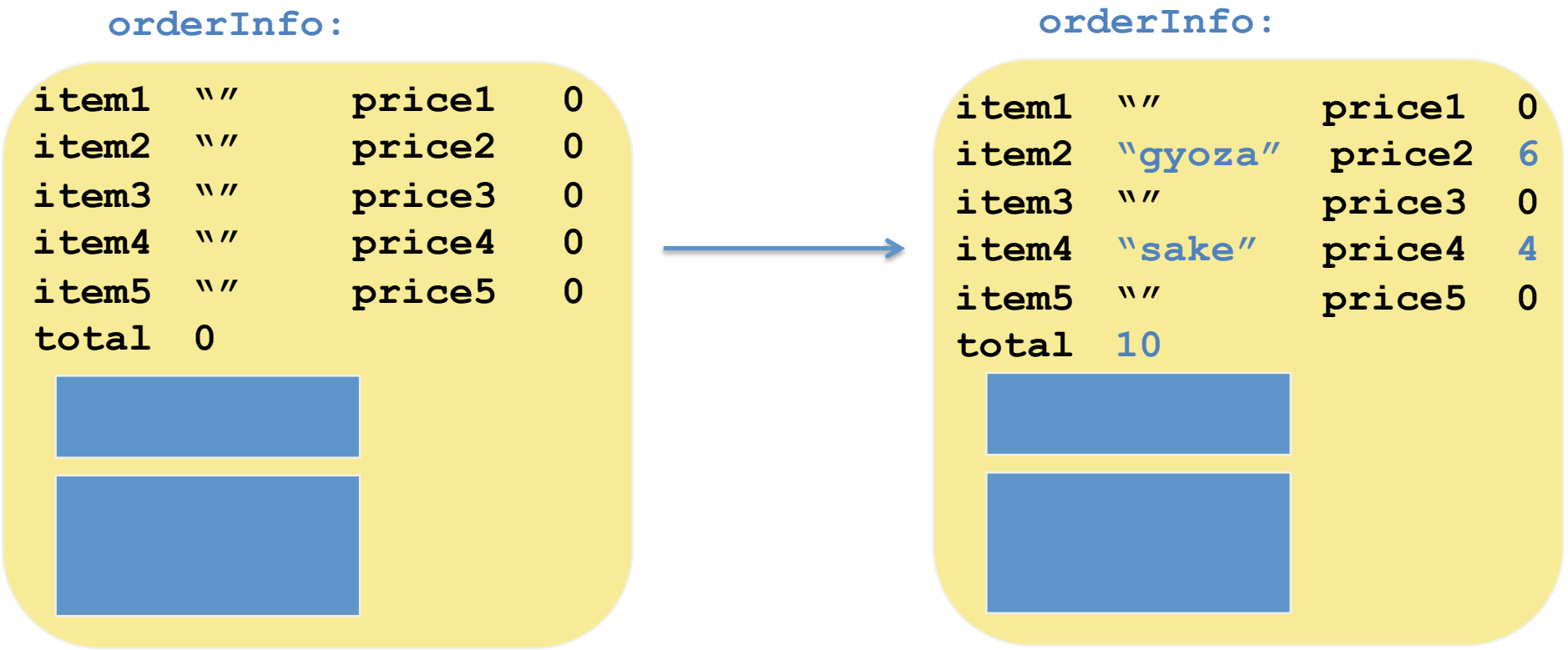
item1	""	price1	0
item2	""	price2	0
item3	""	price3	0
item4	""	price4	0
item5	""	price5	0
total	0		




customerInfo:
name "Jen"
phone 1234567
takeoutOrder null
amountOwing 0
fastFood:
edamame 3
gyoza 6
uniNigiri 5
sakeMaki 4
unagiDon 10
customerInfo
takeoutOrder
Pointers
to objects
orderInfo:
item1 "" price1 0
item2 "" price2 0
item3 "" price3 0
item4 "" price4 0
item5 "" price5 0
total 0

Orders Two Items

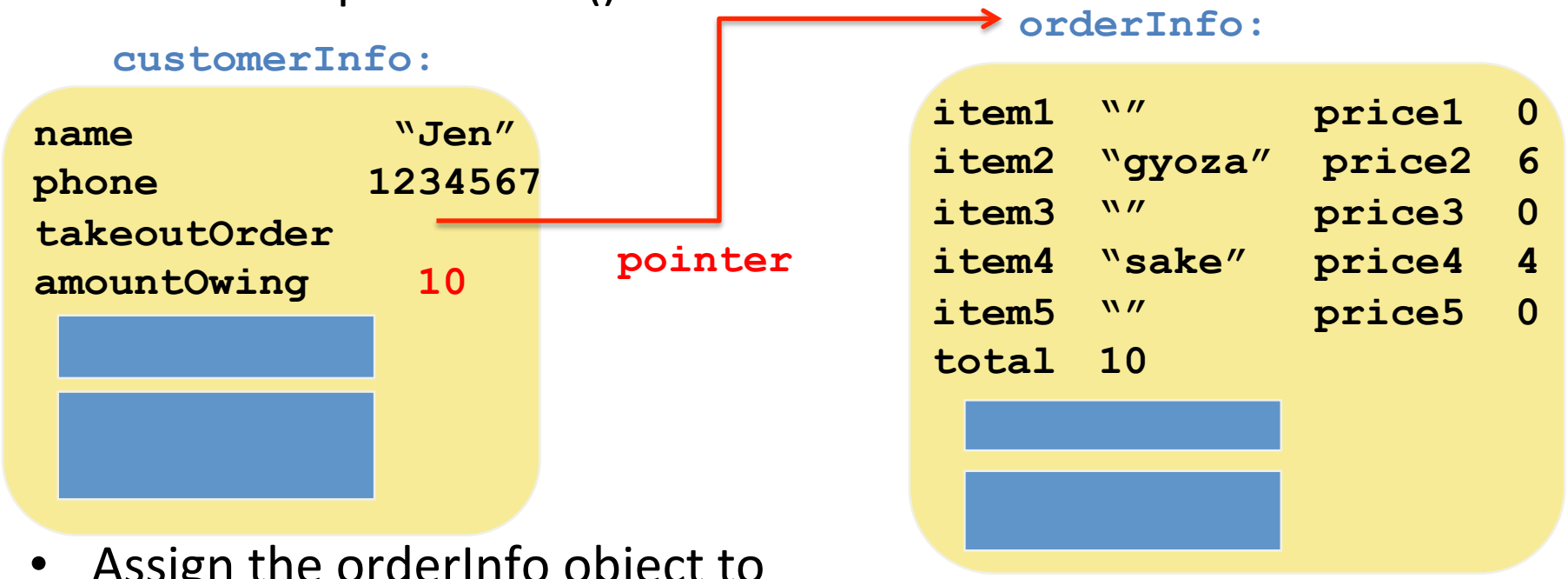
- After ordering gyoza and sake maki:



- No change in customerInfo yet

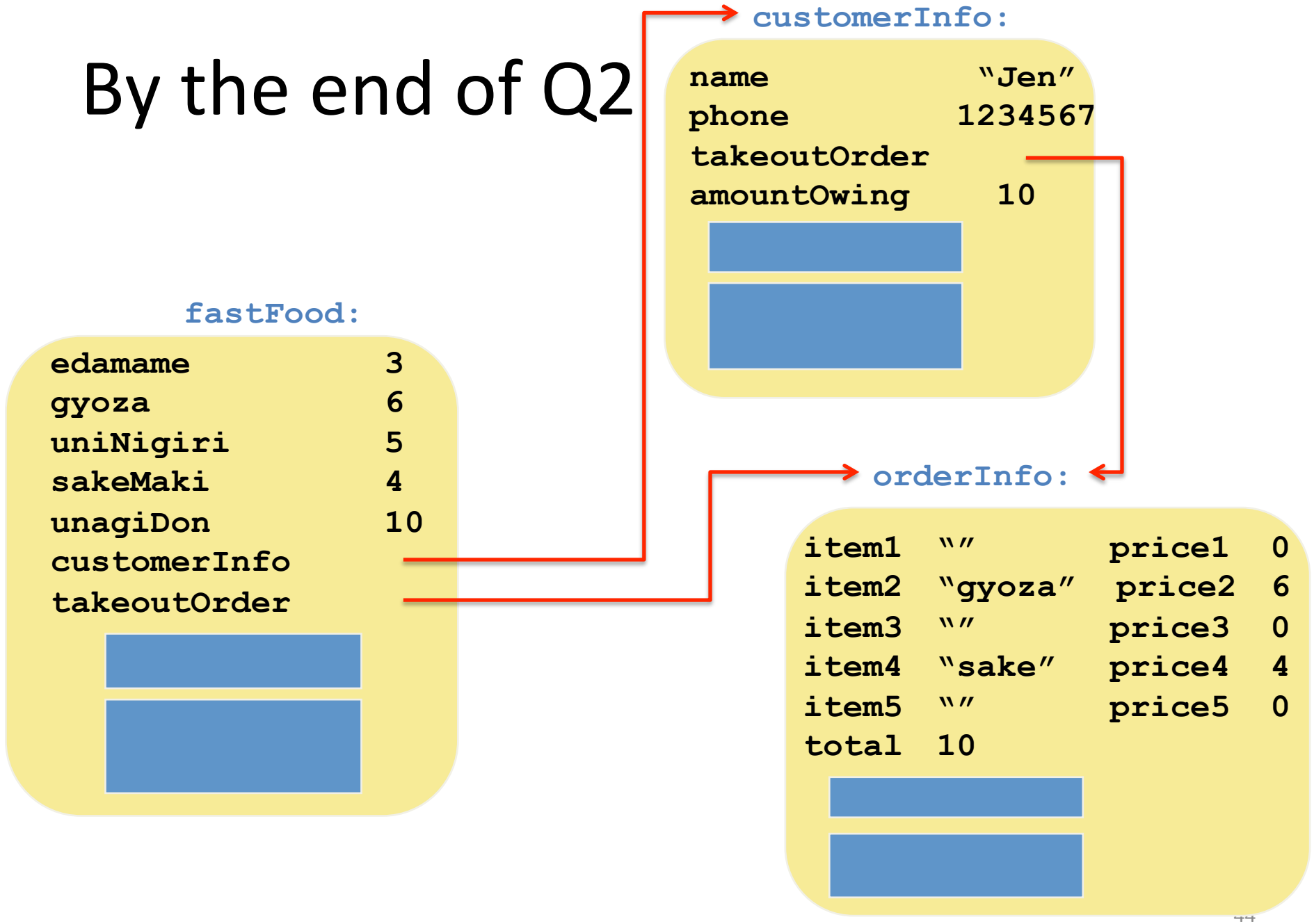
Completing the Order

- When completeOrder() is called:



- Assign the orderInfo object to takeoutOrder (an attribute in the Customer class)
- Assign the total price stored in the orderInfo object to amountOwing (an attribute in the Customer class)

By the end of Q2



By the end of Q3