Lecture 21

- Covers
  - Objects and references
  - Passing parameters with class parameters
- Reading: Savitch 4.3

Lecture overview

- Fundamental difference between simple and object variables
- Difference in assignments
- Difference in comparison tests
- Difference in parameter passing

- Fundamental difference between simple and object variables

Two kinds of variables

- A variable, in its most general sense, can be an attribute, a local variable, or a parameter
- There are 2 kinds of variables:
  - Simple variables (a.k.a. variables of primitive type)
  - Object variables (a.k.a. variables of class type, reference variables, object references)

Fundamental difference

- A simple variable holds the actual value
- An object variable holds the address of the object, not the object itself
- This fundamental difference can cause them to behave differently. In particular, they behave very differently in
  - assignments
  - parameter passing
  - comparison tests

2 remarks

- Why do we have that fundamental difference between the two kinds of variables?
- Don’t confuse object variables with objects

Variables of primitive types

- Name by which we refer to a value of a primitive type
- Stores the actual value
- Often referred to as simple variables

Variables of class type

- Name by which to refer to an object but
- A variable of class type stores the location (memory address) of an object
- The memory address stored is called a reference to an object
- Often referred to as reference variables or object variables

Object references vs objects

- An object reference or object variable is a variable that refers to an object
- An object reference can be an attribute, an argument, or a local variable
- Object references are different from objects
  - In what follows, we have one object but two object references

```
BankAccount x = new BankAccount("123", "Smith");
BankAccount y = x;
```

Object references vs simple variables

- Why the difference?
  - Primitive types store a single value in a fixed amount of memory
  - Objects might be different sizes – e.g. Strings can be any length, and different length strings use different amounts of memory

- Differences in assignments

Differences in assignments

- Compare output from the following code segments

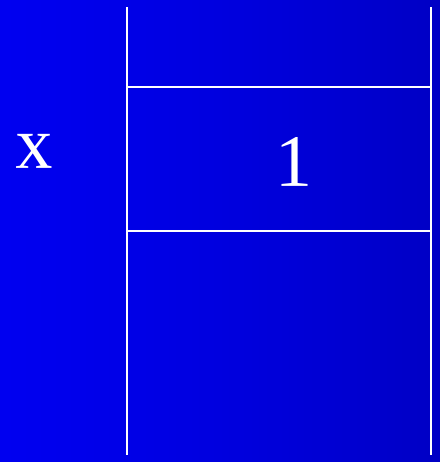
```
int x = 1;
int y = x;
y = 2;
System.out.println( x + " " + y );
```

```
BankAccount x = new BankAccount("123", "Smith");
BankAccount y = x;
y.deposit(100);
System.out.println( x + " " + y );
```

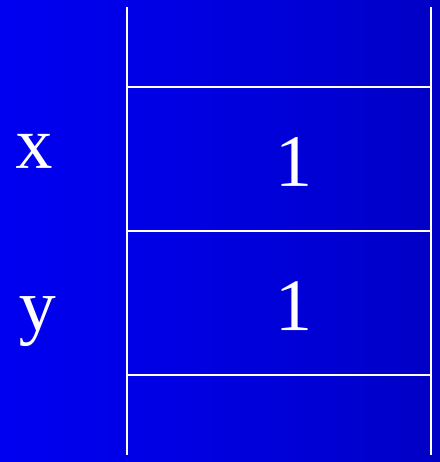
Differences in assignments

- Let x, y be simple variables
- Then the statement y = x;
  - Assigns the value of x to y
  - After that, x and y "operate" independently

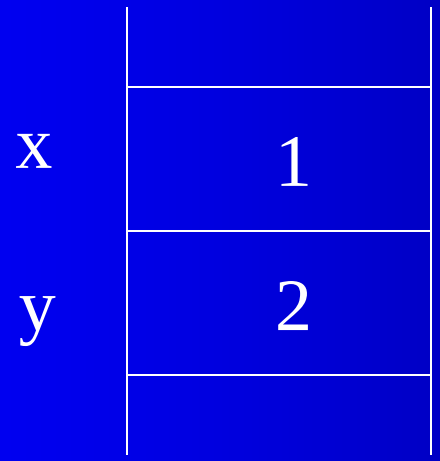
```
int x = 1;
int y = x;
y = 2;
System.out.println( x + " " + y );
```



```
int x = 1;
int y = x;
y = 2;
System.out.println( x + " " + y );
```



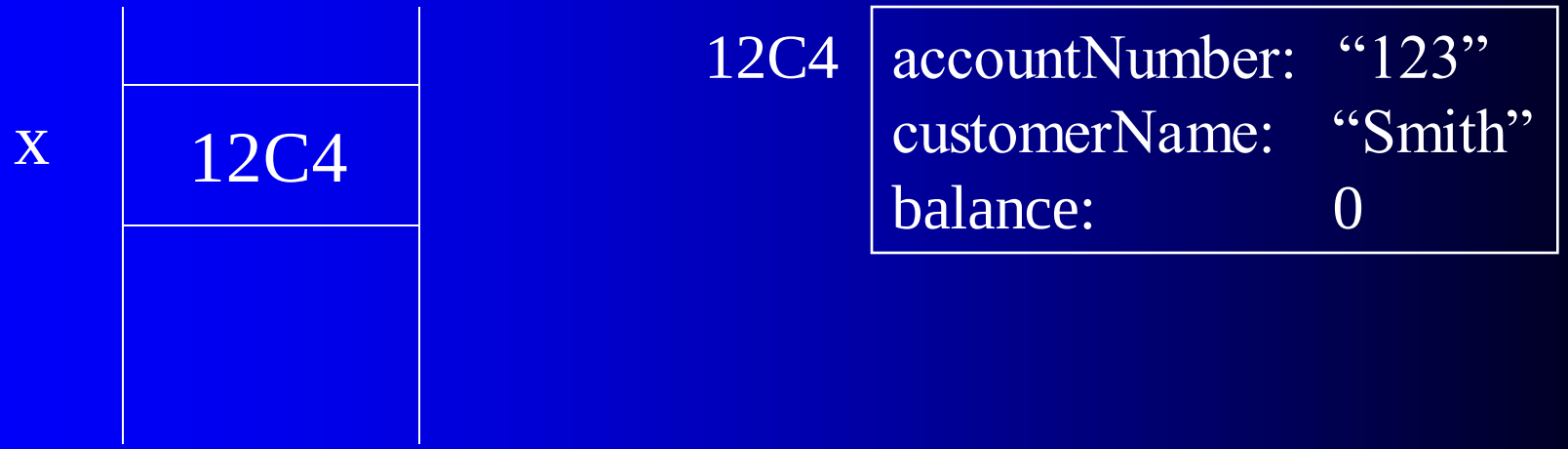
```
int x = 1;
int y = x;
y = 2;
System.out.println( x + " " + y );
```



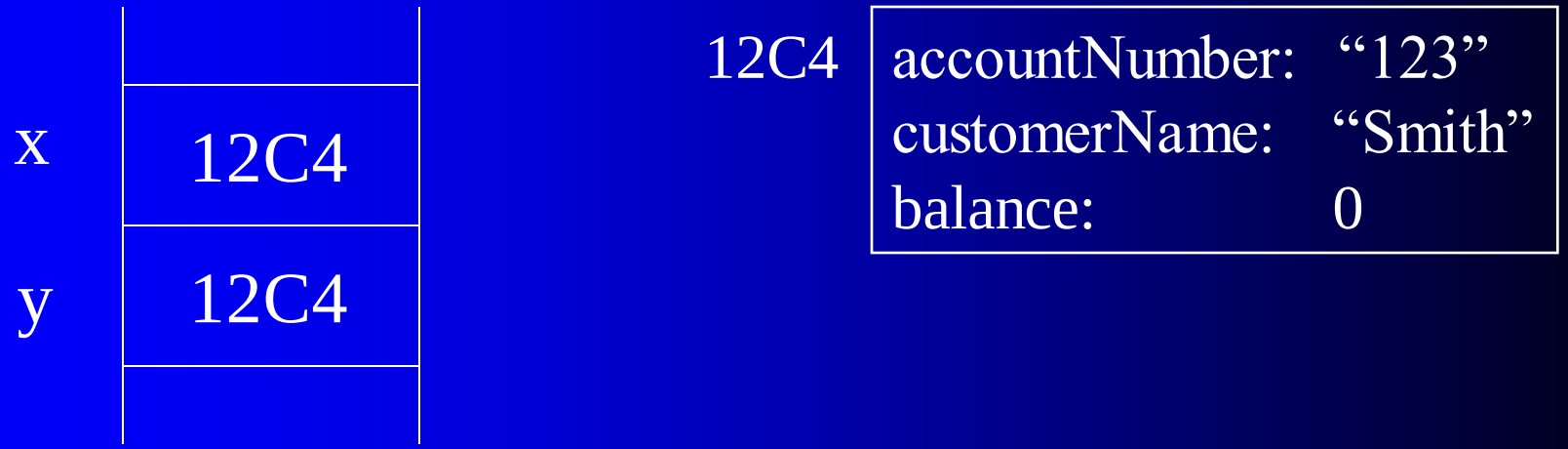
Differences in assignments

- Let x, y be object variables
- Then the statement y = x;
  - Assigns the reference value of x (the address of the object referenced by x) to y
  - Consequently, x and y refer to the same object

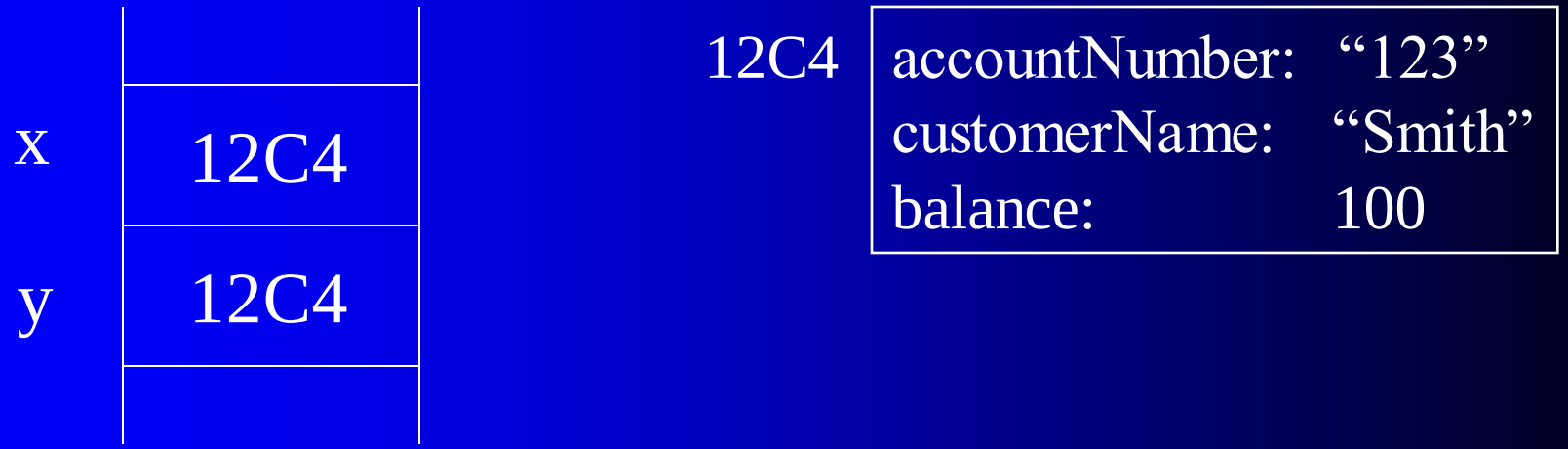
```
BankAccount x = new BankAccount("123", "Smith");
BankAccount y = x;
y.deposit(100);
System.out.println( x + " " + y );
```



```
BankAccount x = new BankAccount("123", "Smith");
BankAccount y = x;
y.deposit(100);
System.out.println( x + " " + y );
```



```
BankAccount x = new BankAccount("123", "Smith");
BankAccount y = x;
y.deposit(100);
System.out.println( x + " " + y );
```



- Differences in comparison tests

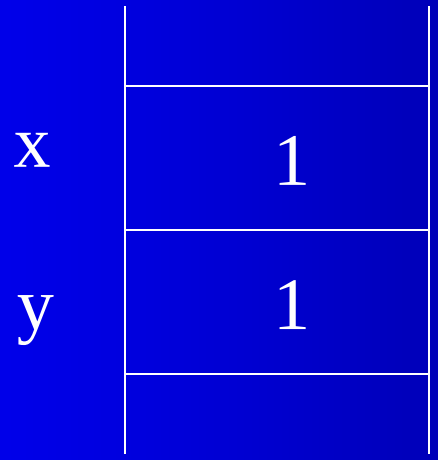
Differences in testing for equality

- Compare the effect of the following code segments

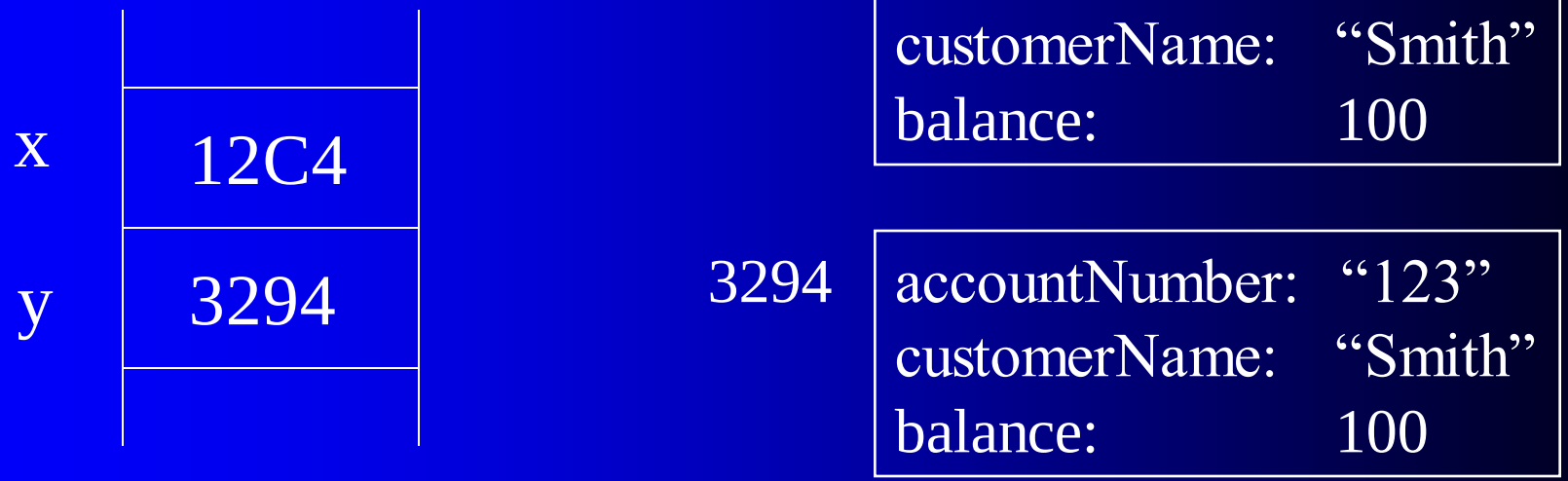
```
int x = 1;
int y = 1;

boolean b = (x == y);
```

```
BankAccount x = new BankAccount("123", "Smith");
BankAccount y = new BankAccount("123", "Smith");

boolean b = (x ==y);
```

```
int x = 1;
int y = 1;

boolean  b = (x == y);
```



```
BankAccount x = new BankAccount("123", "Smith");
BankAccount y = new BankAccount("123", "Smith");

boolean  b = (x ==y);
```



Testing objects for equality

- Always define an equals method to compare the content of two objects
  - Returns a boolean
- To order objects, define a compareTo method
  - Returns an integer

Equals method

- Write an equals method for the BankAccount class

CompareTo method

- Write a compareTo method for the BankAccount class
  - Assume we wish to order BankAccount instances by customer name

- Differences in parameter passing

Differences in parameter passing

- Compare the output of the following code segments

```
int x = 1;
change( x );
System.out.println( "x = " + x );
```

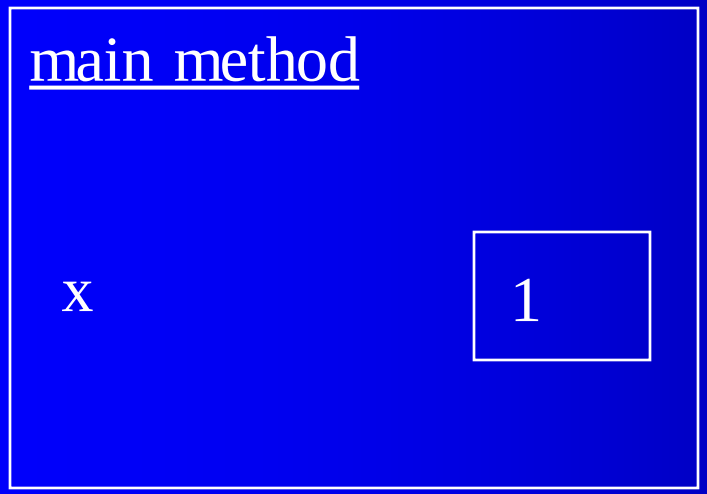
where

```
public void change( int i )
{
    i = i * 2;
}
```

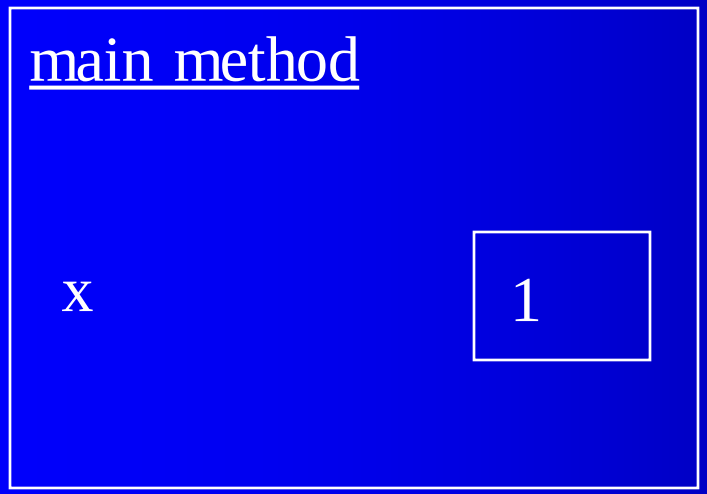
```
int x = 1;
change( x );
System.out.println( "x = " + x );
```

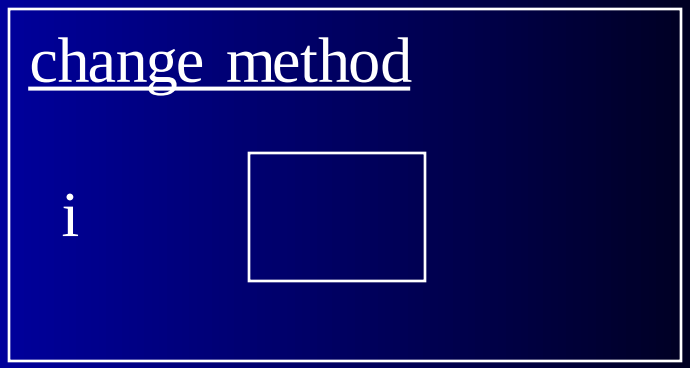
where

```
public void change( int i )
{
   i = i * 2;
}
```

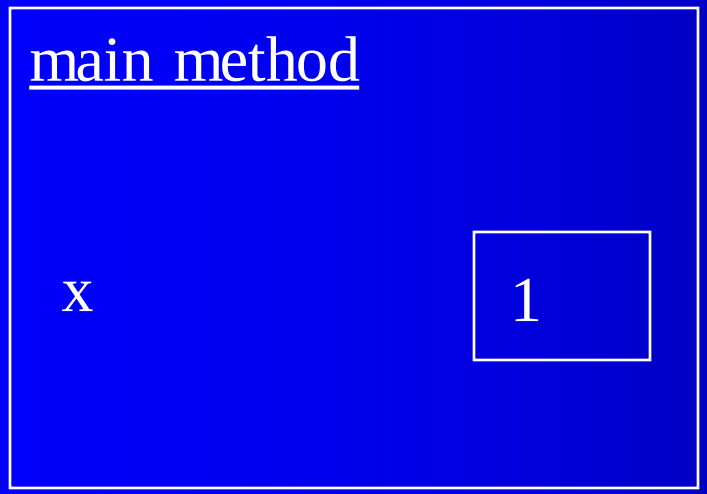


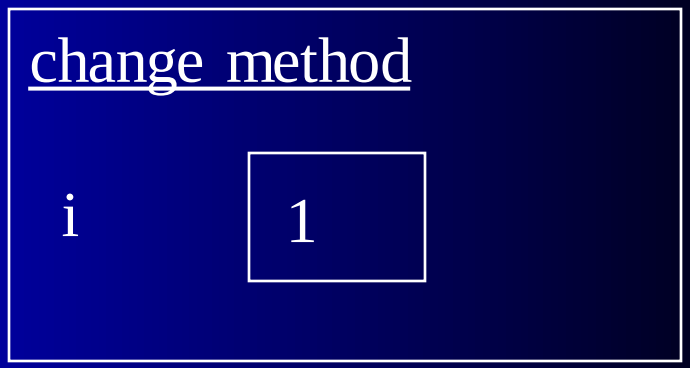
```
  int x = 1;
  change( x );
  System.out.println( "x = " + x );
where
  public void change( int i )
  {
    i = i * 2;
  }
```



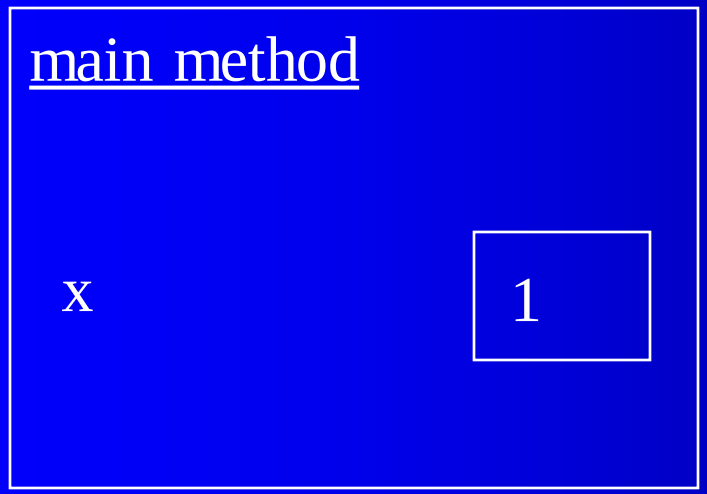


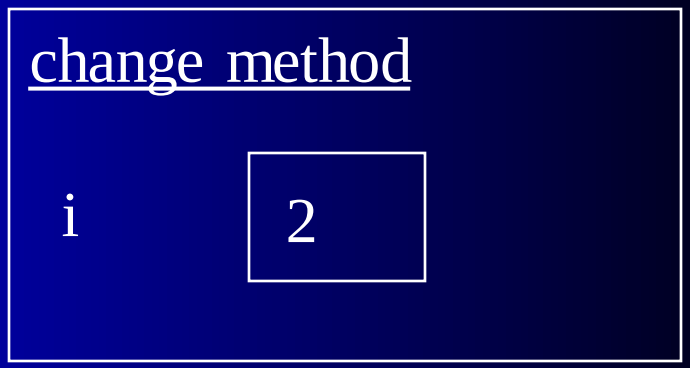
```
  int x = 1;
  change( x );
  System.out.println( "x = " + x );
where
  public void change( int i )
  {
    i = i * 2;
  }
```



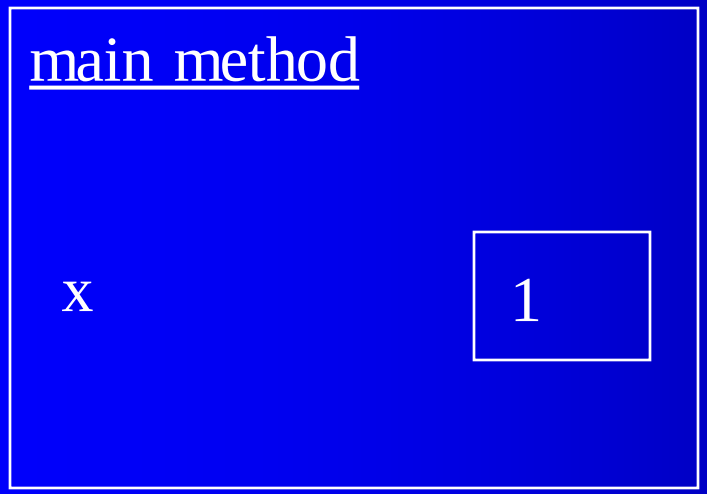


```
  int x = 1;
  change( x );
  System.out.println( "x = " + x );
where
  public void change( int i )
  {
    i = i * 2;
  }
```





```
  int x = 1;
  change( x );
  System.out.println( "x = " + x );
where
  public void change( int i )
  {
    i = i * 2;
  }
```

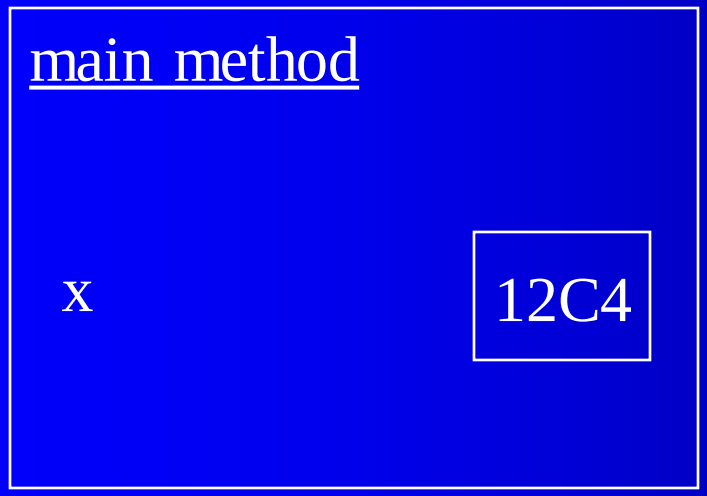


```
BankAccount x = new BankAccount("123", "Smith");
change( x );
System.out.println( "x = "  +  x );
```

where

```
public void  change( BankAccount a  )
{
        a.deposit( 100 );
}
```

```
BankAccount x = new BankAccount("123", "Smith");
change( x );
System.out.println( "x = "  +  x );
where
public void  change( BankAccount a  )
{
   a.deposit( 100 );
}
```



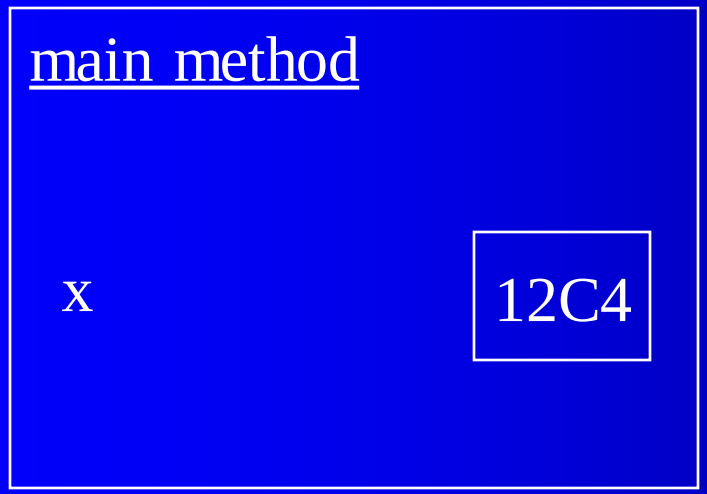
12C4

accountNumber:	“123”
customerName:	“Smith”
balance:	0

```
BankAccount x = new BankAccount("123", "Smith");
change( x );
System.out.println( "x = "  +  x );
```

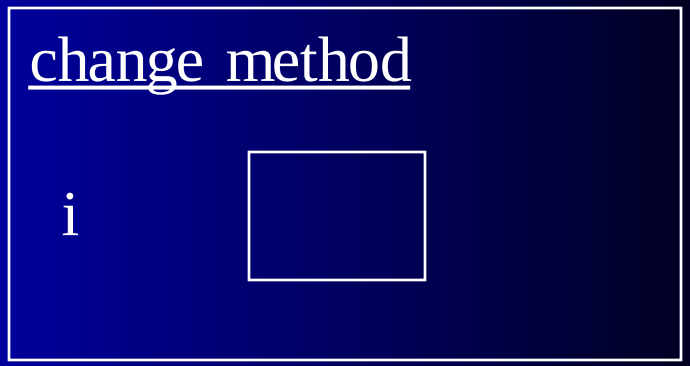
where

```
public void  change( BankAccount a  )
{
   a.deposit( 100 );
}
```



12C4

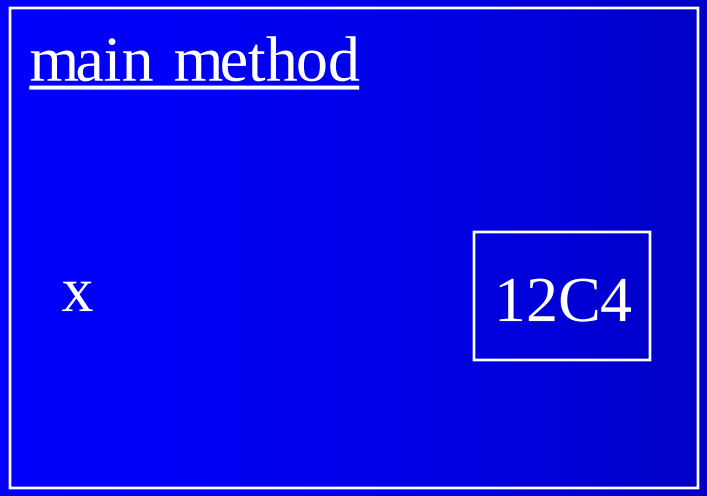
accountNumber: “123”
customerName: “Smith”
balance: 0



```
BankAccount x = new BankAccount("123", "Smith");
change( x );
System.out.println( "x = "  +  x );
```

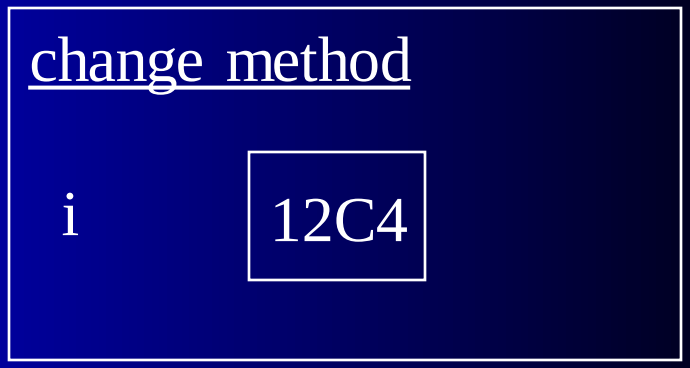
where

```
public void  change( BankAccount a  )
{
   a.deposit( 100 );
}
```

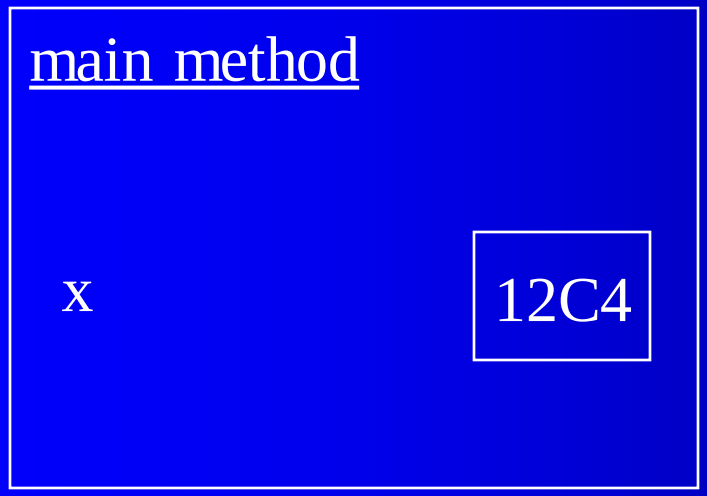


12C4

accountNumber: “123”
customerName: “Smith”
balance: 0

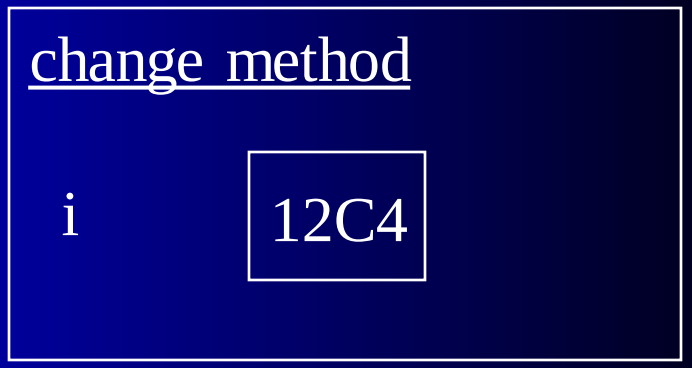


```
BankAccount x = new BankAccount("123", "Smith");
change( x );
System.out.println( "x = "  +  x );
where
public void  change( BankAccount a  )
{
   a.deposit( 100 );
}
```

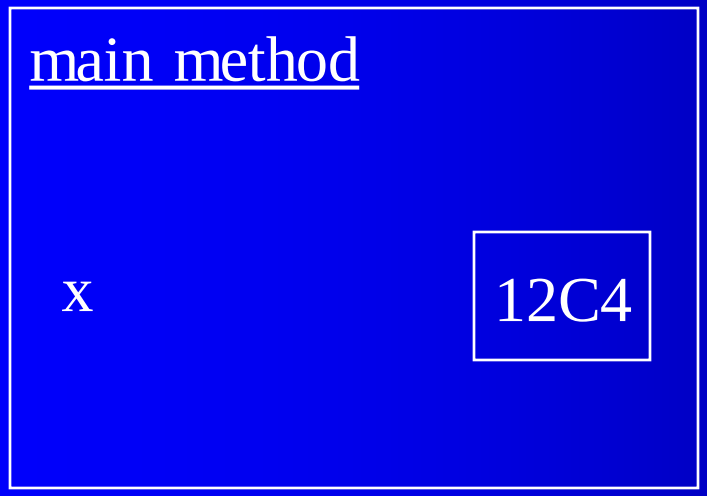


12C4

accountNumber:	“123”
customerName:	“Smith”
balance:	100



```
BankAccount x = new BankAccount("123", "Smith");
change( x );
System.out.println( "x = "  +  x );
where
  public void  change( BankAccount a  )
  {
     a.deposit(  100 );
  }
```



12C4

accountNumber:	“123”
customerName:	“Smith”
balance:	100

Differences in parameter passing

- Let x be a simple variable
- When x is passed to a method
  - The value of x is passed
  - Whatever happens to that value inside the method will not affect the variable x

Differences in parameter passing

- Let x be an object variable
- When x is passed to a method
  - The reference value of x (i.e. the address of the object x refers to) is passed
  - Thus, variable x and the argument point to the same object
  - Therefore, any changes made to the object will be reflected through x as well

A closer look at argument passing

- Given the method

```
public void  exchange( BankAccount a1, BankAccount a2 )
{
  BankAccount temp = a1;
  a1 = a2;
  a2 = temp;
}
```

- What is the effect of the code segment?

```
BankAccount x = new BankAccount("123", "Smith");
BankAccount y = new BankAccount("456", "Jones");
exchange( x, y );
```

Class exercise

- *Q: What is output by this piece of code?*

```
public static void changeValues( int x, double y, String s, DigitalClock b )
{
   x = 99;
   y = 101.1;
   s = "Kangaroo";
   b.setHours(12);
   b.setMinutes(25);
}

…
int a = 0;
double b = 22.2;
String s = "Koala";
DigitalClock dc = new DigitalClock( );
changeValues(a, b, s, dc);
System.out.println( a + " " + b + " " + s + " " + dc);
```

Pass-by-value

- In Java, argument passing is always pass-by-value, i.e. values are passed from the caller to the called method
- For a simple argument, the data value of the actual argument is passed
- For an object argument, the reference value (i.e. the address) of the actual argument is passed

Next lecture

- Programming with methods
- Static attributes
- Static methods
- The Math class