Pass-by-Value

Cheng, Wei  COMP110-001  June 2, 2014
Mid-term

• Grade has been released
Today

• We will discuss in-depth:

1) How primitive values & objects are stored
2) What happens when a method is called
3) How parameters are passed in
What happens...

• when a variable of primitive type is declared & used?

```java
{
    int sum;
    sum = 4;
    sum = sum + 1;
}
```

Memory

Each cell is 1 byte
What happens...

• when a variable of primitive type is declared & used?

```java
{
    int sum;
    sum = 4;
    sum = sum + 1;
}
```
What happens...

• when a variable of primitive type is declared & used?

```java
{ 
    int sum;
    sum = 4;
    sum = sum + 1;
}
```
What happens...

• when a variable of primitive type is declared & used?

```java
{  
    int sum;  
    sum = 4;  
    sum = sum + 1;  
}
```
What happens...

• when a variable of primitive type is declared & used?

```java
{
    int sum;

    sum = 4;

    sum = sum + 1;

} // The scope of variable sum ends here. Memory released
```
Very common mistake in Mid-term

```java
if ( num > 0 ) {
    int absoluteVal = num;
} else {
    int absoluteVal = - num;
}

return absoluteVal; // can you still use this variable here?
```
What happens...

```java
int a = 10;
int b = a;
b = b + 1;

System.out.println( a );
```

What happens....

• When a method with parameter of primitive type is called?

```java
public int abs( int num ) {
    if (num > 0 ) return num;
    else return – num;
}

public void doSth () {
    int someNum = -2;
    System.out.println( abs( someNum ) );
}
```
What happens....

• When a method with parameter of primitive type is called?

```java
public int abs( int num ) {
    if (num > 0 ) return num;
    else return -num;
}
```

```java
public void doSth () {
    int someNum = -2;
    System.out.println( abs( someNum ) );
}
```

This expression is evaluated first. Then the value is copied & passed to the method abs( ... )
What happens....

- When a method with parameter of primitive type is called?

```java
public int abs( int num ) {
    if (num > 0 ) return num;
    else return -num;
}
```

```java
public void doSth () {
    int someNum = -2;
    System.out.println( abs( -2 ) );
}
```

This expression is evaluated first. Then the value is copied & passed to the method abs( ... )
What happens....

• When a method with parameter of primitive type is called?

```java
public int abs( int num ) {
    if (num > 0 ) return num;
    else return -num;
}
```

```java
public void doSth () {
    int someNum = -2;
    System.out.println( abs( -2 ) );
}
```

The value is passed to the method abs( ... )
What happens....

- When a method with parameter of primitive type is called?

public int abs( int num ) {
  if (num > 0 ) return num;
  else return -num;
}

public void doSth () {
  int someNum = -2;
  System.out.println( abs( -2 ) );
}

num is a local variable in the method with value -2.
The value is passed to the method abs( ... )
What happens....

• When a method with parameter of primitive type is called?

You can understand it this way

public int abs(int num) {
    int num = -2;
    if (num > 0) return num;
    else return -num;
}

The value is passed to the method abs( ... )

public void doSth() {
    int someNum = -2;
    System.out.println(abs(-2));
}

num is a local variable in the method with value -2 (the value passed in).
Many mistakes in Mid-term on using method parameters

- When you are writing the method body, use the parameters as local variables with *given* values
- The value is *GIVEN* when the method is *CALLED*
- Do *NOT* try to obtain the value again in some weird way

```java
public int abs( int num ) {
    if (num > 0 ) return num;
    else return – num;
}
public void doSth () {
    int someNum = -2;
    System.out.println( abs( -2 ) );
}
```
What happens....

- When a method with parameter of primitive type is called?

```java
public void increaseByOne(int num) {
    num = num + 1;
}
```

```java
public void doSth() {
    int someNum = -2;
    increaseByOne(someNum);
    System.out.println(someNum);
}
```

What do you get?
What happens....

• When a method with parameter of primitive type is called?

```java
public void increaseByOne( int num ) {
    num = num + 1;
}
```

```java
public void doSth () {
    int someNum = -2;
    increaseByOne( someNum );
    System.out.println( someNum );
}
```

You get the same number: -2!
Why? Because the value -2 is copied and passed in.

int num and int someNum are different variables!
What happens...

• when a variable of Class type is declared & used?

class Student {
    int PID; // 4-byte int
    int year; // 4-byte int
}

....

8 bytes needed for an instance of this class
What happens...

- when a variable of Class type is declared & used?

class Student {
    int PID;
    int year;
}

    Student anna = new Student();
    anna.PID = 1234;
    anna.year = 3;
What happens...

• when a variable of Class type is declared & used?

class Student {
    int PID;
    int year;
}

....

{
    Student anna = new Student();
    anna.PID = 1234;
    anna.year = 3;
}
What happens...

- when a variable of Class type is declared & used?

```java
class Student {
    int PID;
    int year;
}

Student anna = new Student();
anna.PID = 1234;
anna.year = 3;
```

1234 & 3 in binary

Cell 20

How is this address evaluated? The dot resolves address to content
What happens...

• when a variable of Class type is declared & used?

```java
Student anna = new Student();
```

1) An object (orange) is created in memory. Its length depends on the content (members) of the instance

2) The variable `anna` does NOT store the content (orange) directly. It stores the address of the content (green).

Such variables are called references in Java.

Similar to the concept of pointer in C++.
Student anna = new Student();
anna.PID = 1234;
anna.year = 3;

Student a_copy = anna;
a_copy.year = 4;

System.out.println( anna.year );

What do you get?
What happens...

Student anna = new Student();
anna.PID = 1234;
anna.year = 3;

Student a_copy = anna;
a_copy.year = 4;

System.out.println( anna.year );
What happens...

• when a variable of Class type is passed to a method?

```java
{ 
    Student anna = new Student();
    anna.PID = 1234;
    anna.year = 3;
    printStudentInfo( anna );
}

public void printStudentInfo( Student s) {
    System.out.println( s.PID + ":" + s.year );
}
```
What happens...

• when a variable of Class type is passed to a method?

```java
{ 
    Student anna = new Student();
    anna.PID = 1234;
    anna.year = 3;
    printStudentInfo( anna );
}

public void printStudentInfo( Student s ) {
    System.out.println( s.PID + ":" + s.year );
}
```
What happens....

• When a method with parameter of Class type is called?

```java
public void increaseByOne(Student s) {
    s.year = s.year + 1;
}
```

```java
public void doSth() {
    Student anna = new Student();
    anna.PID = 1234;
    anna.year = 3;
    increaseByOne(anna);
    System.out.println(anna.year);
}
```

What do you get?
Central Dogma of Method Calling in Java

• When a method is called:  
  **Pass-by-Value**

• The value of argument is copied and passed in as parameter.
  
  – For primitive types, it’s the value itself
  – For Class types, it’s the address of the actual object content
Next Class

• `== vs .equals()`
• Write your own `.equals()`

• Information hiding