

Week 5

Coding and Using Methods

EXPLORING
Java



This Week

Lecture: More on data types

& Coding and using methods

Java Genesis:

–Ch6: Methods

Quick Quiz for Chapter 5

Primitive types (e.g. integer)



The value of the variable `x` is the integer stored in the location associated with `x`.

trying to swap primitive values

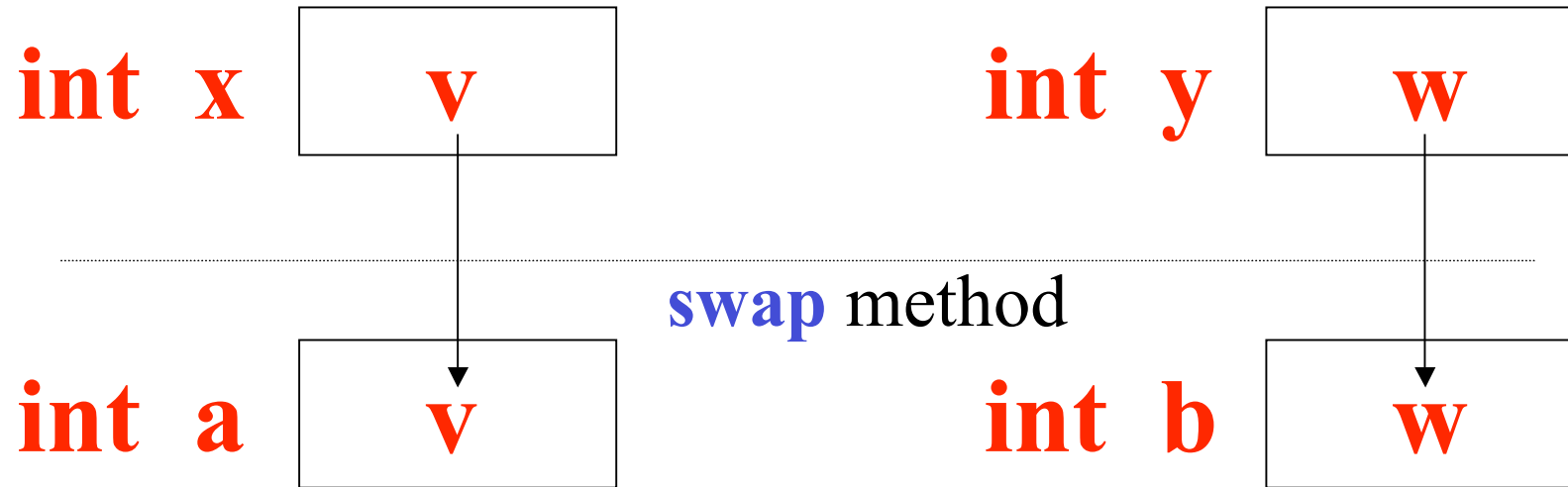
....

```
int x = 5;  
int y = 21;  
swap(x, y);
```

....

```
public static void swap (int a, int b) {  
    int temp = a;  
    a = b;  
    b = temp;  
}
```

passing arguments of primitive type



When `swap` is invoked the integer values of `x` and `y` are copied to new locations. The values in these new locations are then swapped.



trebling primitive values

```
int x = 5;  
treble(x);
```

....

```
public static void treble (int a) {  
    a = 3*a;  
}
```

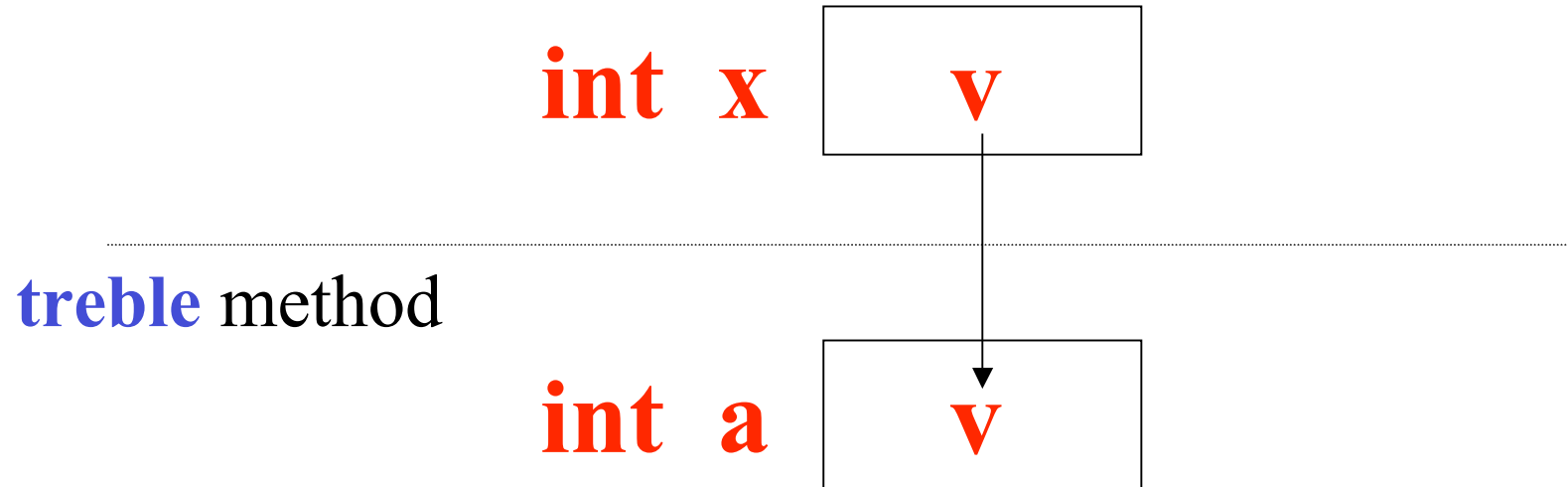
~~--improved code-----~~

```
int x = 5;  
x = treble(x);
```

....

```
public static int treble (int a) {  
    return 3*a;  
}
```

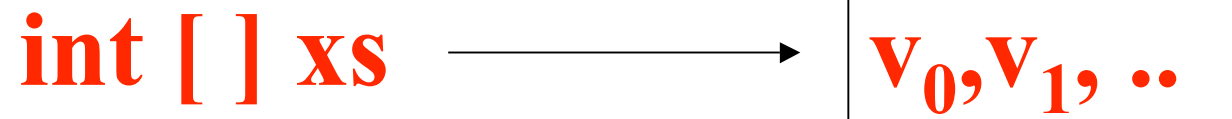
passing arguments of primitive type (cont)



When **treble** is invoked the integer value of `x` is copied to a new location. The value in this new location is then trebled.



Object type (e.g. array)



The value of the variable `xs` is the storage location where the elements (integers) in the array are stored. We can think of `xs` as naming the array.

trebling array values

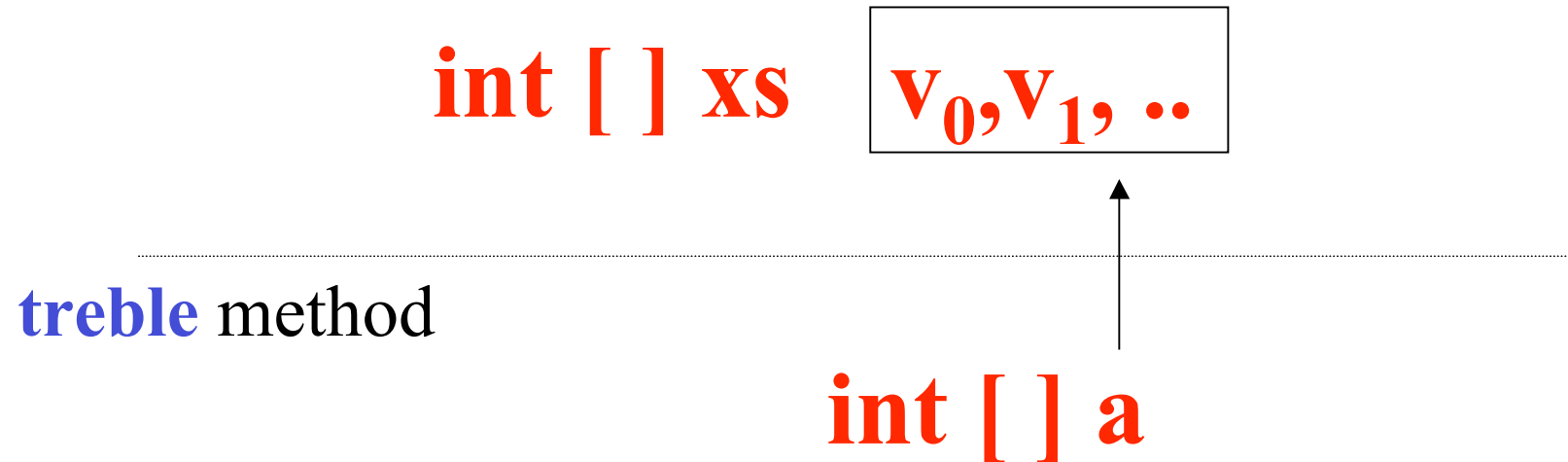
....

```
int [ ] xs = {5, 11, 3};  
treble(xs);
```

....

```
public static void treble (int [ ] a) {  
    for (int i=0; i<a.length; i++) {  
        a[i] = 3*a[i];  
    }  
}
```

passing arguments of object type



When **treble** is invoked location **xs** is now directly referenced (named) by **a**. Hence **a** is an alias for **xs**. Each element in the referenced array is now trebled, so effectively each element in the array **xs** is trebled.

Trying to swap array values

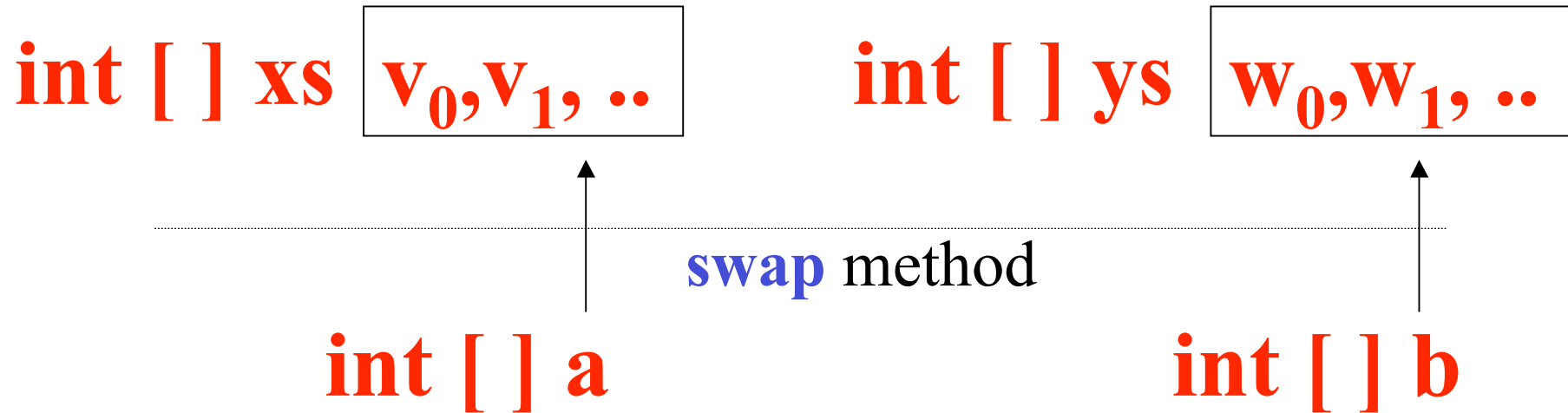
....

```
int [ ] xs = {5, 11, 13};  
int [ ] ys = {4, 10, 12};  
swap(xs, ys);
```

....

```
public static void swap (int [ ] a, int [ ] b) {  
    int [ ] temp = a;  
    a = b;  
    b = temp;  
}
```

passing arguments of object type (cont)



When `swap` is invoked locations `xs` and `ys` are now directly referenced (named) by `a` and `b`. Hence `a` is an alias for `xs`; `b` is an alias for `ys`. If these references are swapped `a` references `ys` and `b` references `xs` but the contents of the arrays are not swapped: they must be swapped element by element.

swapping array values

....

```
int [ ] xs = {5, 11, 13};
```

```
int [ ] ys = {4, 10, 12};
```

```
swap(xs, ys);
```

....

```
public static void swap (int [ ] a, int [ ] b) {
```

```
    int temp;
```

```
    for (int i=0; i<a.length; i++) {
```

```
        temp = a[i];
```

```
        a[i] = b[i];
```

```
        b[i] = temp;
```

```
    }
```

```
}
```

Summing an Array

If a is an array of numbers, then sum of elements in a

$$= a[0] + \text{sum of elements in tail}(a)$$



Problem 1

Generate three random integers as follows:

- first is in the range 1 to 6;
- second is in the range -5 to 5;
- third is in the range 0 to 100.

```
public static int randIntInRange (int from, int to) {  
    int randInt = (int)((to-from+1)*Math.random( )) + from;  
    return randInt;  
}
```



Problem 2

Write a method to find the average of the numbers in an array of integers.

That is, the method takes an array of integers and returns the (floating point) average of the numbers in the array.

```
public static double getAverage (int [ ] xs) {  
    int sum = 0;  
    for (int i=0; i<xs.length; i++) {  
        sum = sum + xs[i];  
    }  
    return (double)sum/xs.length;  
}
```



Problem 3

Write a method to print in the Transcript window all the integers in an array of integers.

That is, the method takes an array of integers and prints in the Transcript window each integer in that array.

```
import genesis.*;
import java.awt.*;
public class Fun {
    public static void printIntArray (int [ ] xs) {
        for (int i=0; i<xs.length; i++) {
            Transcript.print(xs[i] + " ");
        }
        Transcript.println( );
    }
}
```



Problem 4

Write a method to test if all the integers in an array of integers are even.

That is, the method takes an array of integers and returns **true** if all the integers in the array are even, else returns **false**.

```
public static boolean isEven (int [ ] xs) {  
    for (int i=0; i<xs.length; i++) {  
        if (xs[i] % 2 != 0) return false;  
    }  
    return true;  
}
```



Problem 5

Write a method to extract an element at random from an array of integers.

That is, the method takes an array of integers and returns one of the elements in that array selected at random.

```
import genesis.*;
import java.awt.*;
public class Fun {

    public static int randElement(int [ ] xs) {
        int posn = (int)(xs.length*Math.random( ));
        return xs[posn];
    }
}
```



Problem 6

Write a method to extract a colour at random from an array of colours.

That is, the method takes an array of colours and returns one of the colours in that array selected at random.

```
import genesis.*;
import java.awt.*;
public class Fun {

    public static Color randElement(Color [ ] xs) {
        int posn = (int)(xs.length*Math.random( ));
        return xs[posn];
    }
}
```

```

import genesis.*;
import java.awt.*;
public class Fun {
    public static void printIntArray (int [ ] xs) {
        for (int i=0; i<xs.length; i++) {
            Transcript.print(xs[i] + " ");
        }
        Transcript.println( );
    }
    public static int randElement(int [ ] xs) {
        int posn = (int)(xs.length*Math.random( ));
        return xs[posn];
    }
    public static Color randElement(Color [ ] xs) {
        int posn = (int)(xs.length*Math.random( ));
        return xs[posn];
    }
}

```