CIS 110: Introduction to Computer Programming

Lecture 18

Reference semantics

(§ 7.2-7.3)

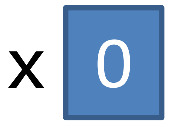
Outline

- Reference semantics
- Array traversals

Reference Semantics

Review: pass by copy

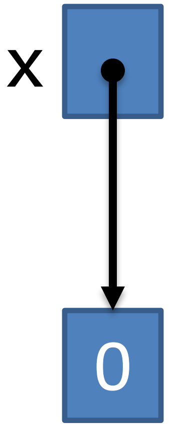
```
public static void change(int x) {
  x = 5;
}

public static void main(String[] args) {
  int x = 0;
  change(x);
  System.out.println(x);
  // Prints 0
}
```



- The x variables are distinct.
- We pass a *copy of the contents* of *x* to change.
- Primitive variables contain their values directly.

The twist: arrays behave differently!

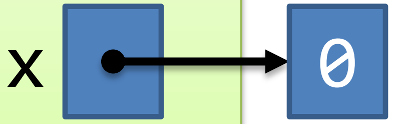
```java
public static void change(int[] x) {
  x[0] = 5;
}

public static void main(String[] args) {
  int x[] = { 0 };
  change(x);
  System.out.println(x[0]);
  // Prints 5
}
```



- Array variables (generally object variables) contain *references* to the arrays rather than the arrays themselves.

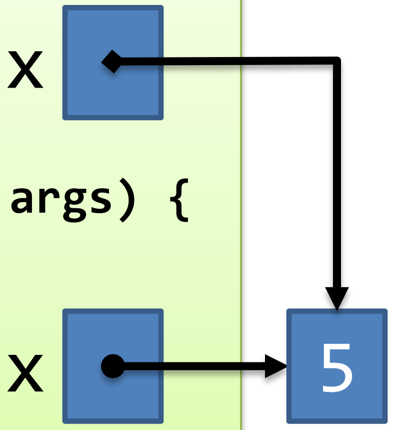
Passing array parameters (1)

```
public static void change(int[] x) {
  x[0] = 5;
}

public static void main(String[] args) {
  int x[] = { 0 };
  change(x);
  System.out.println(x[0]);
  // Prints 5
}
```



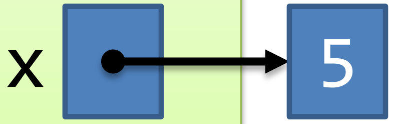
Passing array parameters (2)

```
public static void change(int[] x) {
  x[0] = 5;
}

public static void main(String[] args) {
  int x[] = { 0 };
  change(x);
  System.out.println(x[0]);
  // Prints 5
}
```



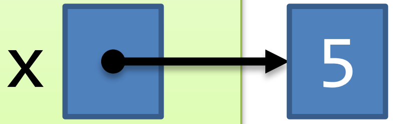
Passing array parameters (3)

```
public static void change(int[] x) {
  x[0] = 5;
}

public static void main(String[] args) {
  int x[] = { 0 };
  change(x);
  System.out.println(x[0]);
  // Prints 5
}
```



Pass by value vs. pass by reference

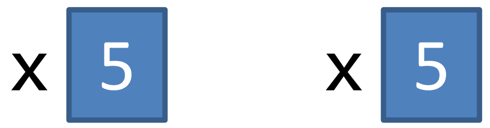
```
public static void change(int[] x) {
  x[0] = 5;
}

public static void main(String[] args) {
  int x[] = { 0 };
  change(x);
  System.out.println(x[0]);
  // Prints 5
}
```

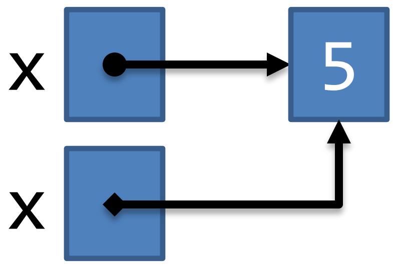


Pass by value vs. pass by reference

- For primitive types we pass *copies* of the contents of the variables.
- For reference types, we pass *references* to the objects the variables refer to.

```
public static void change(int x) {
  x = 5;
}
```



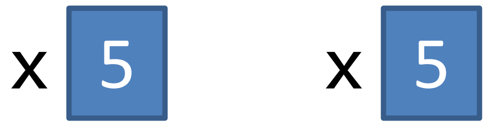


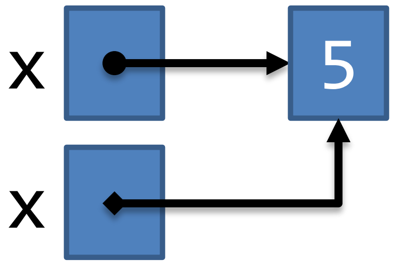
```
public static void change(int[] x) {
  x[0] = 5;
}
```

Alternative view: we copy references

- Alternatively, we always copy the contents of variables along.
  - But we *copy references* of variables of object type.
- Either viewpoint is valid --- pick the one that makes the most sense!

```
public static void change(int x) {
  x = 5;
}
```



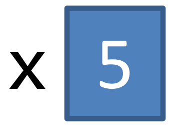


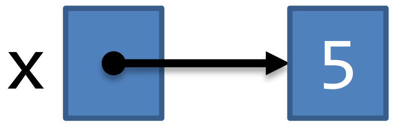
```
public static void change(int[] x) {
  x[0] = 5;
}
```

Alternative view: we copy references

- Alternatively, we always copy the contents of variables along.
  - But we *copy references* of variables of object type.

```java
public static void change(int x) {
  x = 5;
}
```





```java
public static void change(int[] x) {
  x[0] = 5;
}
```

Array Traversals

See Traversals.java, ExtractDigit.java