CS1083 Week 1: Review

Objects, Classes, Arrays

David Bremner

2018-01-08

Syllabus

Numbers and control flow

Objects and classes

Arrays (1D)

Syllabus

Numbers and control flow

Objects and classes

Arrays (1D)

Numbers

What are the values of x and y after the following code is executed:

```
double x = 1.57 + 4 % 7;
int y = ((int)x )/2;
```

if

Which of the following condition statements make no sense?

- if (temp>100 && temp<0)
- if (temp<=100 && temp>=0)
- if (temp>100 || temp<0)
- if (temp>=100 || temp<=0)
- if (temp==100 || temp==0)

if II

If temp has a value of 5, what is the output of the following program?

```
if (temp>0)
System.out.print("Above␣freezing.␣");
    if (temp>30)
        System.out.print("Hot!␣");
else
    System.out.print("Freezing.␣");
```

- Above freezing.
- Above freezing. Freezing.
- Above freezing. Hot!
- Freezing.
- Hot! Freezing.

for loops

What is printed by the following code?

```
int start = -1;
int stop = 1;
for (int k = start; k <= stop; k=k+1) {
  for (int j = k; j <= stop; j=j+1) {
    System.out.print("*");
  }
  System.out.println();
}
```

Syllabus

Numbers and control flow

Objects and classes

Arrays (1D)

Instance variables

What is printed by the following code?

```
public class Test
{
  public int n;
  public static void main(String[] args)
  { int n = 10;
    changeIt(n);
    System.out.println(n);
  }
  public static void changeIt(int i)
  { int n=0;
    i = i * 5;
    n = n+1;
  }
}
```

constructors

Given the following constructor headers:

```
public Thing(String a, int b)
public Thing(String c, double d)
public Thing(String e)
```

which one of the following is invalid?

(a) new Thing("3")

(b) new Thing("B")

(c) new Thing("C", 1)

(d) new Thing("A", 2.2)

(e) new Thing(3)

methods

A *mutator* is

(a) A new superhero movie starring Keanu Reeves

(b) A method to update the value of an object instance variable

(c) A method to return the value of an object instance variable

(d) A special instance variable that changes in response to button clicks

Syllabus

Numbers and control flow

Objects and classes

Arrays (1D)

Why arrays?

Prices1

```
Scanner sc = new Scanner(System.in);
double price1, price2, price3, price4, price5, …, price17;
price1 = sc.nextDouble();
price2 = sc.nextDouble();
price3 = sc.nextDouble();
price4 = sc.nextDouble();
price5 = sc.nextDouble();
⋮
price17 = sc.nextDouble();
```

Problems with this approach

- Tedious.
- What if there are 18 prices?

Why arrays?

Prices1

```
Scanner sc = new Scanner(System.in);
double price1, price2, price3, price4, price5, …, price1
price1 = sc.nextDouble();
price2 = sc.nextDouble();
price3 = sc.nextDouble();
price4 = sc.nextDouble();
price5 = sc.nextDouble();
⋮
price17 = sc.nextDouble();
```

Prices1

```
min=price1;
if (price2 < min)
        min=price2;
if (price3 < min)
        min=price3;
⋮
if (price17 < min)
        min=price17;
```

Why arrays?

```
min=price1;
if (price2 < min)
        min=price2;
if (price3 < min)
        min=price3;
⋮
if (price17 < min)
        min=price17;
```

Prices1

Problems with this approach

- Tedious.
- What if there are 18 prices?

creating and using arrays

Prices2

```
1  int numPrices=17;
2  double [] price=new double [numPrices];
3  for (int i=0; i<price.length; i++){
4      price[i]=sc.nextDouble();
5  }
6  double min=price[0];
7  for (int i=1; i<numPrices; i++){
8      if (price[i]<min)
9          min=price[i];
10 }
11 System.out.println(min);
```

Interesting lines

2 Declaring an array

3 The length is available as an “instance variable”

4 Using an array in a loop

6 Arrays start from 0

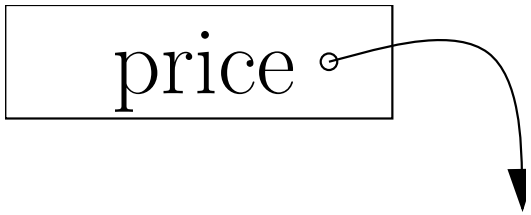
Step by step: Declaring arrays

```
double price[]=new double[17];
```

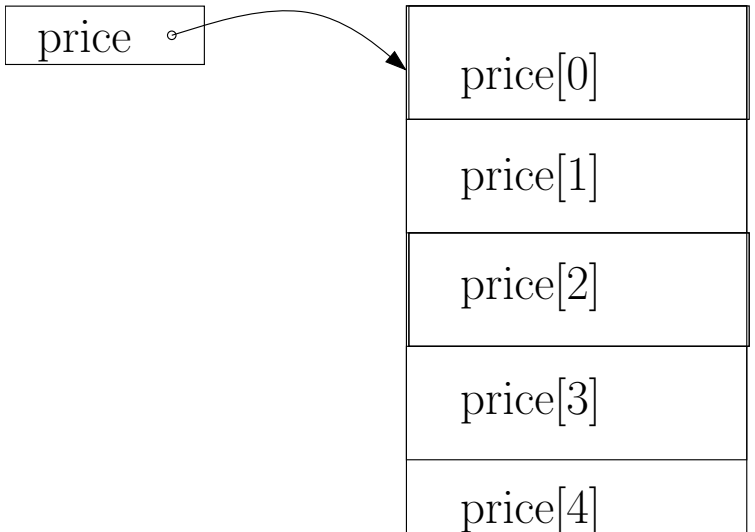
this is actually short for

```
1 double price[];
2 price=new double[17];
```

1. declares an *array reference*

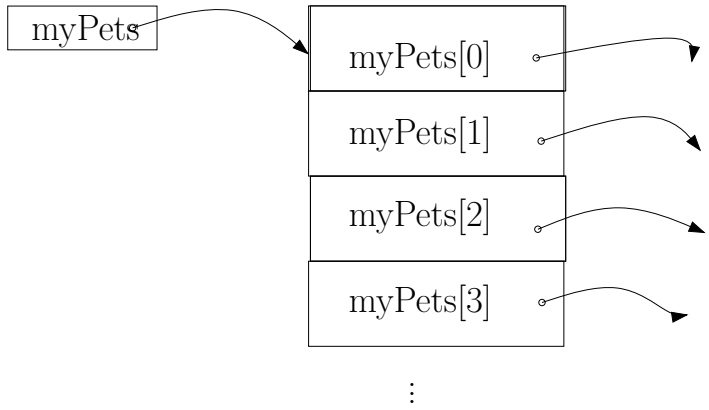


2. creates an *array object* and stores its address in variable `price`

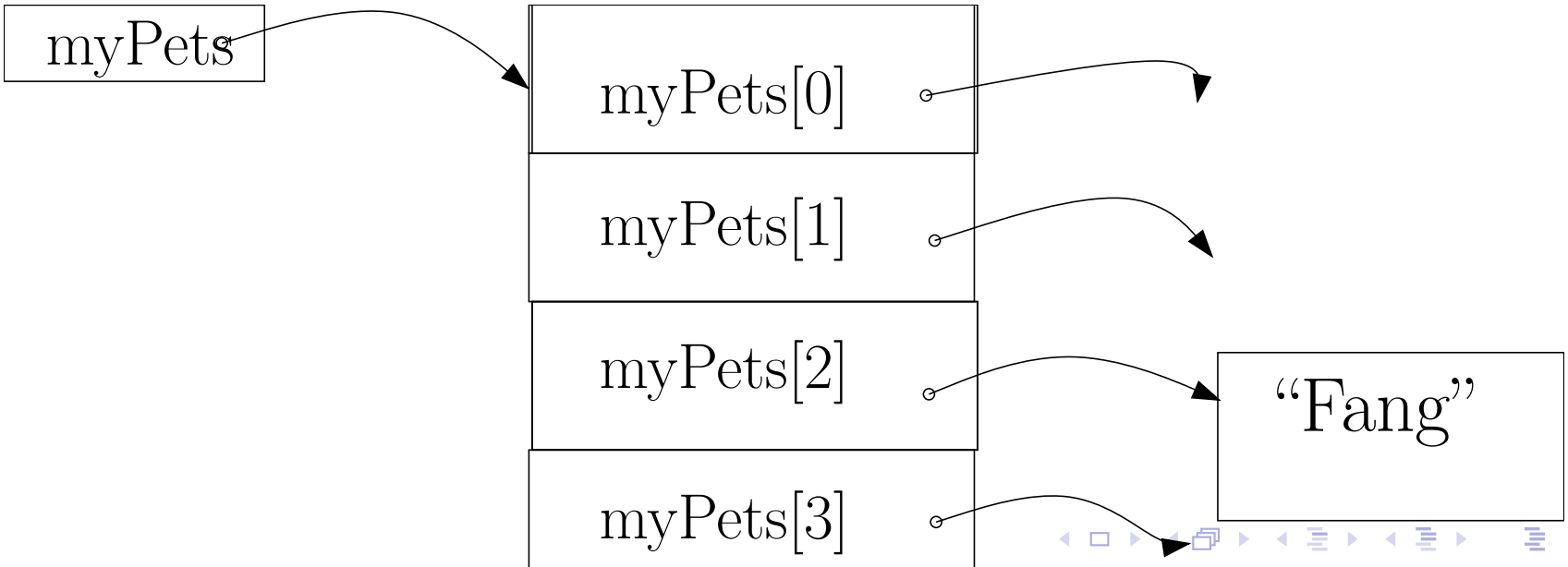


Declaring arrays of objects

```
CyberPet[] myPets=new CyberPet[17];
```




```
myPets[2]=new CyberPet("Fang");
```



using arrays

Array Element `price[i]`

- `arrayName [ indexExpression ]`
- `indexExpression` evaluates to an integer
- $0 \leq$ `indexExpression` $\leq$ `arrayName.length`

Use

Anywhere a variable of the base type (reference or primitive), e.g.

- On either side of an assignment
  - `price[i]=Double.parseDouble(in.readline());`
  - `min=price[i];`
- In an `if`

$$\begin{array}{l} \textit{if } (price[i]<min) \\ \vdots \end{array}$$

Kinds of Java Variables

primitive types

numbers int, double , float, double, char

- `chars` are numbers?!?

boolean

object references `CyberPet`

array references

- Like objects
  - created with `new`
  - can have many *references* to the same array or object
- Not like objects
  - No
  - Exactly one (final) instance variable
  - They have the "index" [] operator.

Kinds of Java Variables

primitive types

numbers int, double , float, double, char

- `chars` are numbers?!?

boolean

object references `CyberPet`

array references

- Like objects
  - created with `new`
  - can have many *references* to the same array or object
- Not like objects
  - No methods
  - Exactly one (final) instance variable
  - They have the "index" [] operator.