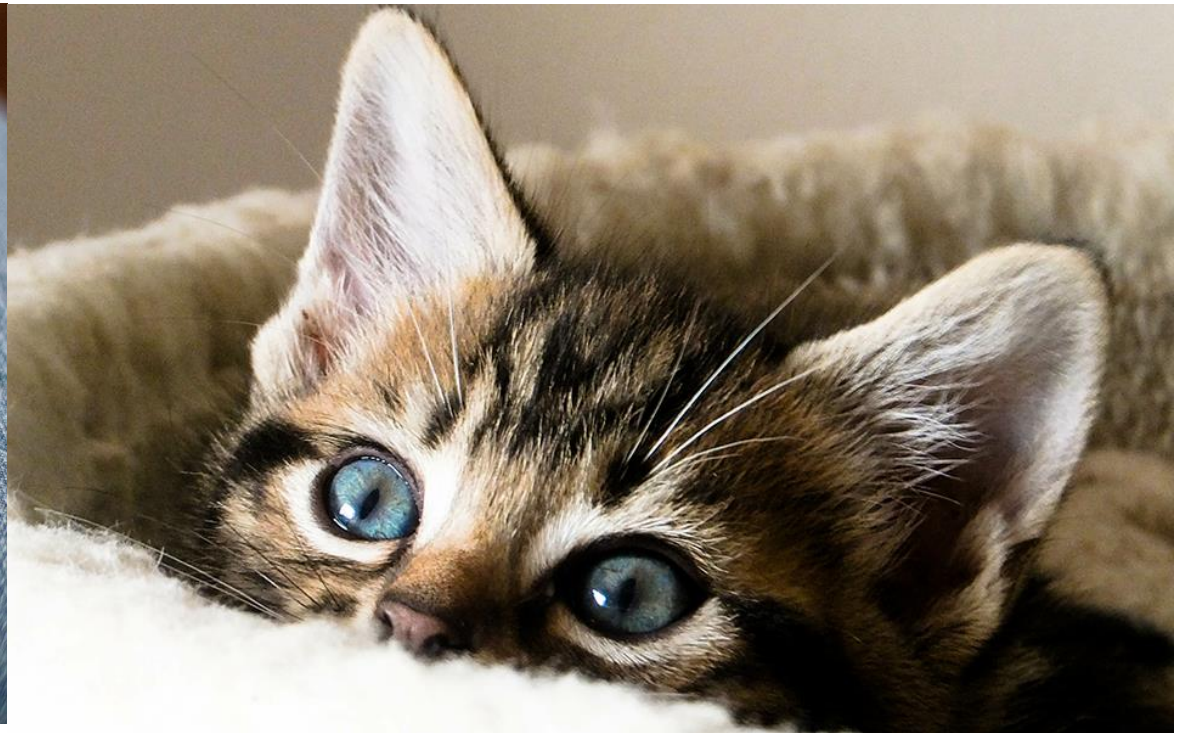


Baby Objects Q & A

Everything all together all at once



Method Type?

```
public class Item {  
    double price; 1  
    String name;
```

```
public Item() {  
    price = 13.45; 2  
    name = "t-shirt";
```

```
public Item(double p, String n) {  
    price = p; 3  
    name = n;
```

```
public double getPrice() { 4  
    return price;
```

```
public String getName() {  
    return name;
```

```
public String toString() { 5  
    return "The " + name + " costs $" + price;
```

```
public void setPrice(double p) { 6  
    price = p;
```

```
public void setName (String n) { 7  
    name = n;
```

```
public boolean equals(Item i) {  
    if (i.getName().equals(name)  
        && i.getPrice() == price)  
        return true;
```

```
else 8  
    return false;
```

```
public int compareTo(Item i) {
```

```
    //on the basis of price
```

```
    if (i.getPrice() > price) 9  
        return -1;
```

```
    else if (i.getPrice() == price)  
        return 0;
```

```
    else  
        return 1;
```

```
}}
```

Instance Variables:

- The variables that you want to store for your object.
- Your object will group these variables together into a new complex type

```
public class Item {  
    private double price;  
    private String name;
```

Begin the class

Declare the
instance
variables

First Line
of Class
File

Class name:
ClassName

Instance Variable:
double InsVar

public

1

2

{

First Line
of Class
File

Class name:
ClassName

Instance Variable:
double InsVar

public

class

1

ClassName

2

}

First Line
of Class
File

Class name:
Triangle

Instance Variable:
int base

public

1

2

{

First Line
of Class
File

Class name:
Triangle

Instance Variable:
int base

public

class

1

Triangle

2

}

First Line
of Class
File

Class name:
Dog

Instance Variable:
String name

public

1

2

}

First Line
of Class
File

Class name:
Dog

Instance Variable:
String name

public

class

1

Dog

2

}

Constructors

- Initialize and set up memory.
- You need a default AND one with parameters for each instance variable.

```
public Item() {  
    price = 13.45;  
    name = "t-shirt";  
}
```

Default.
Put a value in
each instance
variable.

```
public Item(double p, String n) {  
    price = p;  
    name = n;  
}
```

Take each parameter,
assign to instance
variable.

Parameter for
each instance
variable.

```
public Item() {  
    price = 13.45;  
    name = "t-shirt";  
}
```

Constructors are special.

- They have no return type because the type they return is themselves (in this case, an Item).
- They must have the same name as the class.
- When they are called, they are called with the word `new` and the class name.

```
Item shoe = new Item(23.45, "flip-flops");  
Item shirt = new Item();
```

Default
Constructor

Class name:
ClassName

Instance Variable:
double InsVar

public

1

2

3

4

return
type

method
name

param
type

param
name

5

=

6

;

}

{

Default
Constructor

Class name:
ClassName

Instance Variable:
double InsVar

public

ClassName

---blank---

1 return type 2 method name 3 param type 4 param name

InsVar

3.14159

Any useful
default value

}

Default
Constructor

Class name:
Book

Instance Variable:
boolean isFiction

public

1

2

3

4

return
type

method
name

param
type

param
name

5

=

6

;

}

}

Default
Constructor

Class name:
Book

Instance Variable:
boolean isFiction

public

Book

---blank---

1 return type 2 method name 3 param type 4 param name

isFiction

=

false

;

5

6

Any useful
default value

}

}

Default
Constructor

Class name:
Person

Instance Variable:
char firstInitial

public

1

2

3

4

return
type

method
name

param
type

param
name

5

=

6

;

}

Default
Constructor

Class name:
Person

Instance Variable:
char firstInitial

public

Person

---blank---

1 return type 2 method name 3 param type 4 param name

firstInitial

'a'

Any useful
default value

}

5

6

Customized Constructor

Class name:
ClassName

Instance Variable:
double InsVar

```
public 1 2 (3 4) {  
    5 = 6 ;  
}
```

return type method name param type param name

Customized Constructor

Class name:
ClassName

Instance Variable:
double InsVar

public

ClassName

double

I

1 return
type

2 method
name

3 param
type

4 param
name

InsVar

=

I

;

5

6

parameter

First letter of
instance
variable

}

Customized Constructor

Class name:
Circle

Instance Variable:
int radius

```
public 1 2 (3 4) {  
    5 = 6 ;  
}
```

return type method name param type param name

Customized Constructor

Class name:
Circle

Instance Variable:
int radius

public

Circle

int

r

1 return
type

2 method
name

3 param
type

4 param
name

radius

=

r

;

5

6

parameter

First letter of
instance
variable

}

Customized Constructor

Class name:
Horse

Instance Variable:
String name

```
public 1 2 (3 4) {  
    5 = 6 ;  
}
```

return type method name param type param name

Customized Constructor

Class name:
Horse

Instance Variable:
String name

public

Horse

String

n

1

return
type

2

method
name

3

param
type

4

param
name

name

=

n

;

5

6

parameter

First letter of
instance
variable

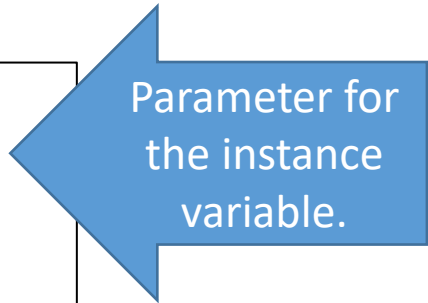
}

}

Mutators

- Change memory
- You need one for each instance variable.

```
public void setPrice (double p) {  
    price = p;  
}  
  
public void setName (String n) {  
    name = n;  
}
```



Parameter for
the instance
variable.



Take parameter,
assign to the right
instance variable.

Mutator

Class name:
ClassName

Instance Variable:
double InsVar

public

1

2

3

4

return
type

method
name

param
type

param
name

5

=

6

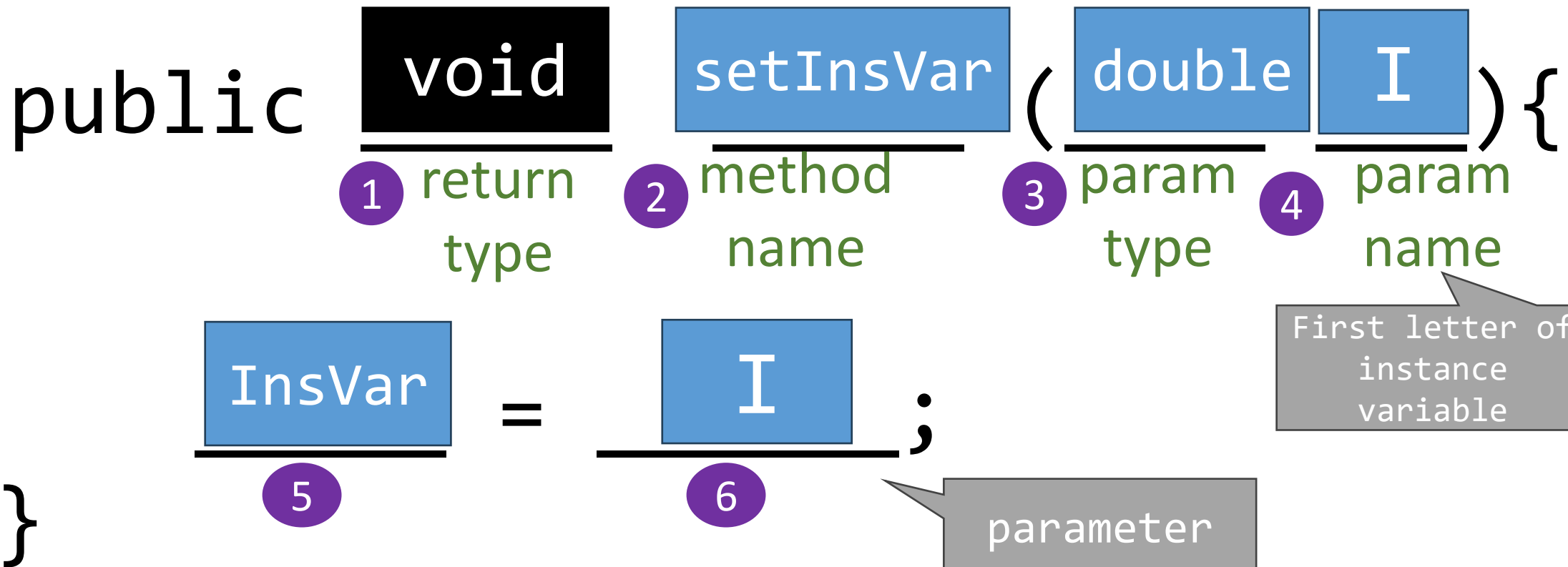
;

}

Mutator

Class name:
ClassName

Instance Variable:
double InsVar



Mutator

Class name:
Turtle

Instance Variable:
String species

public

1

return
type

2

method
name

3

param
type

4

param
name

5

=

6

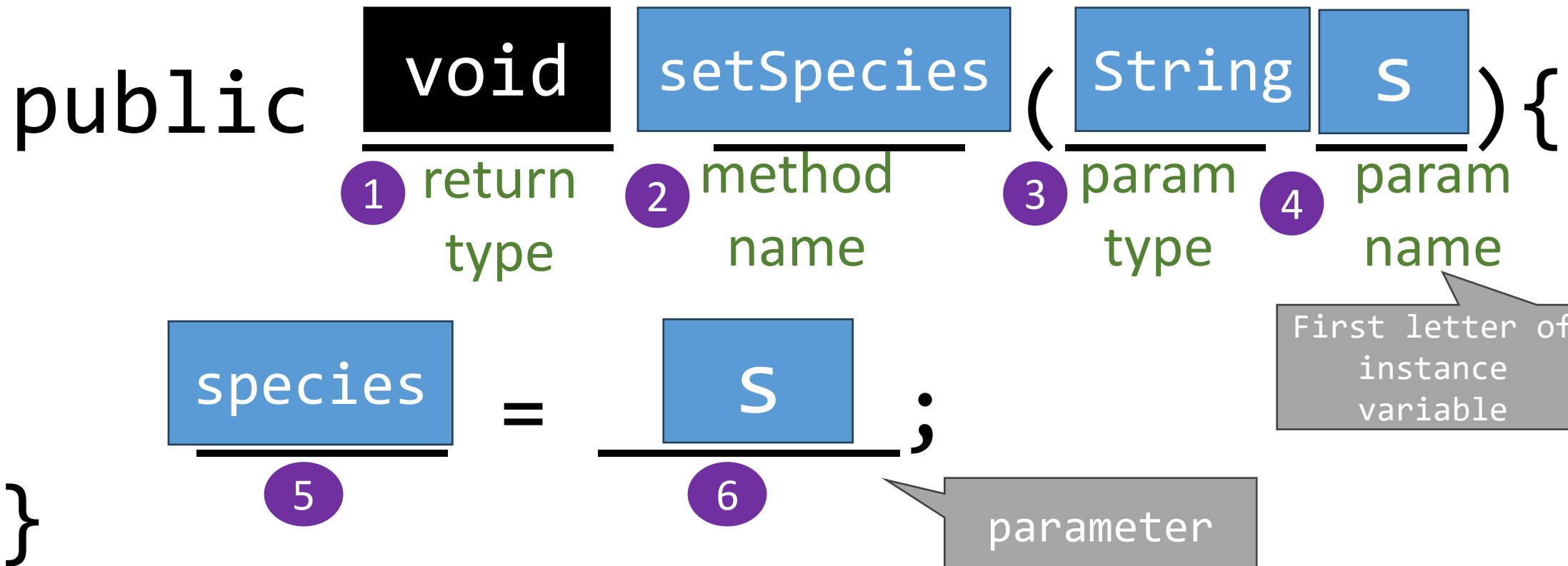
;

}

Mutator

Class name:
Turtle

Instance Variable:
String species



Mutator

Class name:
cube

Instance Variable:
int length

public

1

2

3

4

return
type

method
name

param
type

param
name

5

=

6

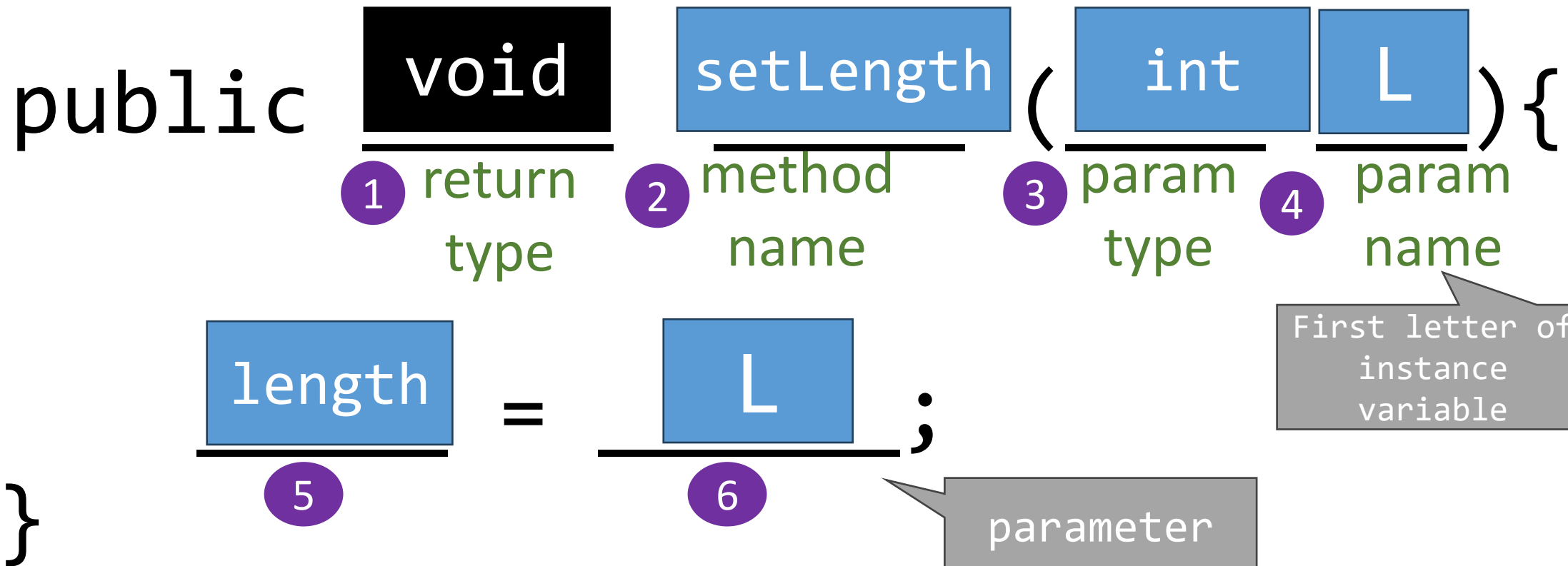
;

}

Mutator

Class name:
cube

Instance Variable:
int length



Accessors

- Access what is stored in memory.
- You need one for each instance variable.

```
public double getPrice () {  
    return price;  
}
```

Return type matches the instance variable type.

```
public String getName () {  
    return name;  
}
```

Return correct instance variable

```
public String toString () {  
    return "The "+name+" costs $"+price;  
}
```

Make a sentence out of the variables.

Accessor

Class name:
ClassName

Instance Variable:
double InsVar

public

1

2

3

4

return
type

method
name

param
type

param
name

5

6

}

Accessor

Class name:
ClassName

Instance Variable:
double InsVar

```
public double getInsVar ( --blank-- ) {  
    return InsVar ;  
}
```

1 return type

2 method name

3 param type

4 param name

5

Type of instance variable

Accessor

Class name:
Fruit

Instance Variable:
char colour

public

1

2

3

4

return
type

method
name

param
type

param
name

5

6

}

Accessor

Class name:
Fruit

Instance Variable:
char colour

public

char

getColour

--blank--

1

return
type

2

method
name

3

param
type

4

param
name

Type of
instance
variable

return

colour

5

}

Accessor

Class name:
Animal

Instance Variable:
int age

public

1

2

3

4

return
type

method
name

param
type

param
name

5

6

}

Accessor

Class name:
Animal

Instance Variable:
int age

public

int

getAge

--blank--

1

return
type

2

method
name

3

param
type

4

param
name

Type of
instance
variable

return

age

5

}

ToString

Class name:
ClassName

Instance Variable:
double InsVar

public

1

2

3

4

return
type

method
name

param
type

param
name

5

6

}

;

ToString

Class name:
ClassName

Instance Variable:
double InsVar

```
public String toString ( --blank-- ) {  
    return "Value is:" + InsVar ;  
}
```

1 return type 2 method name 3 param type 4 param name

5 6

String with words and instance variables

ToString

Class name:
Tree

Instance Variable:
double height

public

1

2

3

4

return
type

method
name

param
type

param
name

5

6

}

;

ToString

Class name:
Tree

Instance Variable:
double height

```
public String toString ( --blank-- ) {  
    return "Value is:" + height ;  
}
```

1 return type 2 method name 3 param type 4 param name

5 6

String with words and instance variables

ToString

Class name:
House

Instance Variable:
String postalCode

public

1

2

3

4

return
type

method
name

param
type

param
name

5

6

}

;

ToString

Class name:
House

Instance Variable:
String postalCode

```
public String toString ( --blank-- ) {  
    return "Value is:" + postalCode ;  
}
```

1 return type 2 method name 3 param type 4 param name

5 6

String with words and instance variables

Facilitator: Equals

- Sees if two of your new type are equal

Pass in an object
that is the same
type as your class

```
public boolean equals (Item i) {  
    if (i.getName ().equals (name)  
        && i.getPrice () == price)  
        return true;  
    else  
        return false;  
}
```

For each instance variable,
see if it matches the
parameters' value

Return true if all instance
variables match, false
otherwise.

Equals

Class name:
ClassName

Instance Variable:
double InsVar

```
public 1 2 (3 4) {  
    if (5 == 6 . 7)  
        return 8 ;  
    else  
        return 9 ;  
}
```

return type

method name

param type

param name

Equals

Class name:
ClassName

Instance Variable:
double InsVar

```
public boolean equals (ClassName C) {  
    if (InsVar == C.getInsVar())  
        return true;  
    else  
        return false;  
}
```

1 return type
2 method name
3 param type
4 param name
5
6
7
8
9

Equals

Class name:
Matrix

Instance Variable:
int rows

```
public 1 2 (3 4) {  
    if (5 == 6 . 7)  
        return 8 ;  
    else  
        return 9 ;  
}
```

return type method name param type param name

Equals

Class name:
Matrix

Instance Variable:
int rows

```
public boolean equals (Matrix M) {  
    if (row == M.getRows())  
        return true;  
    else  
        return false;  
}
```

1 return type
2 method name
3 param type
4 param name
5
6
7
8
9

Equals

Class name:
Equation

Instance Variable:
double slope

```
public 1 2 (3 4) {  
    if (5 == 6 . 7)  
        return 8 ;  
    else  
        return 9 ;  
}
```

return type

method name

param type

param name

Equals

Class name:
Equation

Instance Variable:
double slope

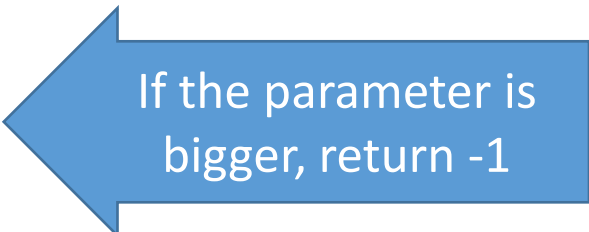
```
public boolean equals (Equation E) {  
    if ( slope == E.getSlope() )  
        return true ;  
    else  
        return false ;  
}
```

1 return type
2 method name
3 param type
4 param name
5
6
7
8
9

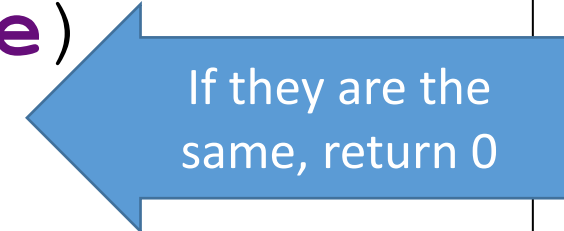
Facilitator: CompareTo

- Sees how two of your new type compare, for sorting

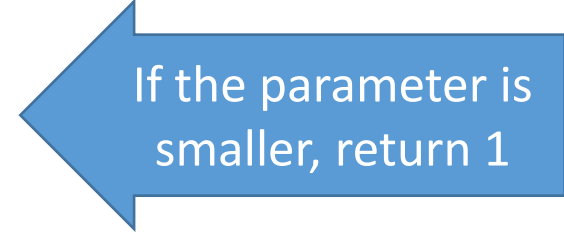
```
public int compareTo (Item i) {  
    //on the basis of price  
    if (price>i.getPrice ())  
        return 1;  
    else if (i.getPrice ()==price)  
        return 0;  
    else  
        return -1;  
}
```



If the parameter is bigger, return -1



If they are the same, return 0



If the parameter is smaller, return 1

CompareTo

Class name:
ClassName

Instance Variable:
double InsVar

```
public 1 2 (3 4) {  
    if (5 > 6 . 7)  
        return 8 ;  
    else if (9 < 10 . 11)  
        return 12 ;  
    else  
        return 13 ;  
}
```

CompareTo

Class name:
ClassName

Instance Variable:
double InsVar

```
public int compareTo (ClassName C) {  
    if (InsVar > C.getInsVar())  
        return 1;  
    else if (InsVar < C.getInsVar())  
        return -1;  
    else  
        return 0;  
}
```

1 return type 2 method name 3 p type 4 param name

Me > Them

I win

Me < Them

They win

Me == Them

We tie

13

CompareTo

Class name:
Tax

Instance Variable:
double percent

```
public 1 2 (3 4) {  
    if (5 > 6 . 7)  
        return 8 ;  
    else if (9 < 10 . 11)  
        return 12 ;  
    else  
        return 13 ;  
}
```

return type method name param type param name

CompareTo

Class name:
Tax

Instance Variable:
double percent

```
public int compareTo ( Tax T ) {
```

1 return type 2 method name 3 p type 4 param name

Me > Them

```
if ( percent > T.getPercent() )
```

I win

```
return 1 ;
```

Me < Them

```
else if ( percent < T.getPercent() )
```

They win

```
return -1 ;
```

Me == Them

```
else  
return 0 ;
```

We tie

```
}
```

13

CompareTo

Class name:
Ques

Instance Variable:
char ans

```
public 1 2 (3 4) {  
    if (5 > 6 . 7)  
        return 8 ;  
    else if (9 < 10 . 11)  
        return 12 ;  
    else  
        return 13 ;  
}
```


CompareTo

Class name:
Ques

Instance Variable:
char ans

```
public int compareTo ( Ques Q ) {
```

1 return type 2 method name 3 p type 4 param name

Me > Them

```
if ( ans > Q.getAns() )
```

```
return 1 ;
```

I win

Me < Them

```
else if ( ans < Q.getAns() )
```

```
return -1 ;
```

They win

Me == Them

```
else  
return 0 ;
```

We tie

```
}
```

13