References, classes and objects

Andy Guest
a.guest@yorksj.ac.uk

27 January 2019

Welcome to 1CB105 Programming 2

- My name is Andy Guest, my email is `a.guest@yorksj.ac.uk`

Welcome to 1CB105 Programming 2

- My name is Andy Guest, my email is `a.guest@yorksj.ac.uk`
- My office is in 44 Lord Mayor's Walk

Today

- Introduction to the module.

Today

- Introduction to the module.
- Object-oriented programming.

Today

- Introduction to the module.
- Object-oriented programming.
- Classes and member variables.

Today

- Introduction to the module.
- Object-oriented programming.
- Classes and member variables.
- References.

Welcome to 1CB105 Programming 2

In this module we will:

Welcome to 1CB105 Programming 2

In this module we will:

- Learn about object-oriented programming:

Welcome to 1CB105 Programming 2

In this module we will:

- Learn about object-oriented programming:
  - What it is.

Welcome to 1CB105 Programming 2

In this module we will:

- Learn about <u>object-oriented programming</u>:
  - What it is.
  - Why everyone is always going on about it.

Welcome to 1CB105 Programming 2

In this module we will:

- Learn about <u>object-oriented programming</u>:
  - What it is.
  - Why everyone is always going on about it.
  - Why it's good.

Welcome to 1CB105 Programming 2

In this module we will:

- Learn about <u>object-oriented programming</u>:
  - What it is.
  - Why everyone is always going on about it.
  - Why it's good.
  - How to do it in Java.

Welcome to 1CB105 Programming 2

In this module we will:

- Learn about <u>object-oriented programming</u>:
  - What it is.
  - Why everyone is always going on about it.
  - Why it's good.
  - How to do it in Java.
- Learn about <u>graphical user interfaces</u>:

Welcome to 1CB105 Programming 2

In this module we will:

- Learn about <u>object-oriented programming</u>:
  - What it is.
  - Why everyone is always going on about it.
  - Why it's good.
  - How to do it in Java.
- Learn about <u>graphical user interfaces</u>:
  - How to make windows, buttons, sliders etc. in Java.

Welcome to 1CB105 Programming 2

In this module we will:

- Learn about <u>object-oriented programming</u>:
  - What it is.
  - Why everyone is always going on about it.
  - Why it's good.
  - How to do it in Java.
- Learn about <u>graphical user interfaces</u>:
  - How to make windows, buttons, sliders etc. in Java.
  - As a bit of light relief.

Expectations

- Listen when I am talking to the class (I won't talk too much!)

Expectations

- Listen when I am talking to the class (I won't talk too much!)
- Keep the room calm when we are working on exercises.

Expectations

- Listen when I am talking to the class (I won't talk too much!)
- Keep the room calm when we are working on exercises.
- Do the exercises from each week before the next week's lecture.

Expectations

- Listen when I am talking to the class (I won't talk too much!)
- Keep the room calm when we are working on exercises.
- Do the exercises from each week before the next week's lecture.
- Ask for help if you get stuck or you're not sure how something works.

Timetable

- Monday: lecture/lab (9am–1pm or 2pm–6pm).

Timetable

- Monday: lecture/lab (9am–1pm or 2pm–6pm).
- Tuesday: SOL (9am–12pm and 2pm–5pm).

How to do well in this module

- Practice

How to do well in this module

- Practice, practice

How to do well in this module

- Practice, practice, practice

How to do well in this module

- Practice, practice, practice, then practice more.

How to do well in this module

- Practice, practice, practice, then practice more.
- If you write code you will get good at writing code.

How to do well in this module

- Practice, practice, practice, then practice more.
- If you write code you will get good at writing code.
- If you don't write code you will (probably) not get good at writing code.

How to do well in this module

- Practice, practice, practice, then practice more.
- If you write code you will get good at writing code.
- If you don't write code you will (probably) not get good at writing code.
- If you are confused — ask.

How to do well in this module

- Practice, practice, practice, then practice more.
- If you write code you will get good at writing code.
- If you don't write code you will (probably) not get good at writing code.
- If you are confused — ask.
- If you don't understand something, the next thing probably isn't going to make much sense either.

How to do well in this module

- Practice, practice, practice, then practice more.
- If you write code you will get good at writing code.
- If you don't write code you will (probably) not get good at writing code.
- If you are confused — ask.
- If you don't understand something, the next thing probably isn't going to make much sense either.
- I can normally explain things in more than one way, so try me!

Object-oriented programming (OOP)

There are two parts to learning about this.

Object-oriented programming (OOP)

There are two parts to learning about this.

- How do I do it.

Object-oriented programming (OOP)

There are two parts to learning about this.

- How do I do it.
- Why should I do it?

Object-oriented programming (OOP)

There are two parts to learning about this.

- How do I do it.
- Why should I do it?

I am going to show you:

Object-oriented programming (OOP)

There are two parts to learning about this.

- How do I do it.
- Why should I do it?

I am going to show you:

- How to do it.

Object-oriented programming (OOP)

There are two parts to learning about this.

- How do I do it.
- Why should I do it?

I am going to show you:

- How to do it.
- How it can make programs easier to read, write and think about.

A simple class

```
class Foo {
  int x;
  int y;
}

Foo f = new Foo ();
```

A simple class

```
class Foo {
  int x;
  int y;
}

Foo f = new Foo ();
```

What just happened?

A simple class

```
Foo f = new Foo();
```

A simple class

```
Foo f = new Foo();
```


Create a new object...

A simple class

...using a plan (or `class') called Foo...

```
Foo f = new Foo();
```

Create a new object...

A simple class

...using a plan (or `class') called Foo...

```
Foo f = new Foo();
```

...and give it the label f

Create a new object...

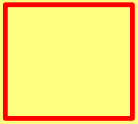
A simple class

```
Foo f = new Foo();
```

A simple class

```
Foo f = new Foo();
```

Create a new object...



A simple class

...using a plan (or `class') called Foo...

```
Foo f = new Foo();
```

Create a new object...

```
x = 0
y = 0
```

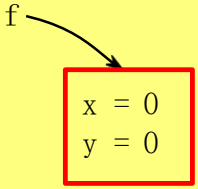
A simple class

...using a plan (or `class') called Foo...

```
Foo f = new Foo();
```

...and give it the label f

Create a new object...



A simple class — changing the object

```
class Foo {
  int x;
  int y;
}

Foo f = new Foo ();
f. x = 42;
f. y = 9;
```

A simple class — changing the object

```
class Foo {
  int x;
  int y;
}

Foo f = new Foo ();
f. x = 42;
f. y = 9;
```

What just happened?

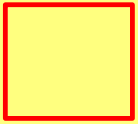
A simple class — changing the object

```
Foo f = new Foo();
```

A simple class — changing the object

```
Foo f = new Foo();
```

Create a new object...



A simple class — changing the object

...using a plan (or `class') called Foo...

```
Foo f = new Foo();
```

Create a new object...

```
x = 0
y = 0
```

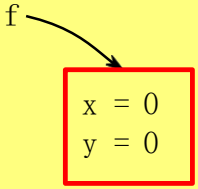
A simple class — changing the object

...using a plan (or `class') called Foo...

```
Foo f = new Foo();
```

...and give it the label f

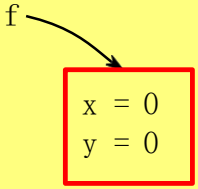
Create a new object...



A simple class — changing the object

Look in the object labelled f

```
Foo f = new Foo();
f.x = 42;
```



A simple class — changing the object

Look in the object labelled f

```
Foo f = new Foo();
f.x = 42;
```

Find its x variable

f

```
x = 0
y = 0
```

A simple class — changing the object

Look in the object labelled f

```
Foo f = new Foo();
f.x = 42;
```

Find its x variable

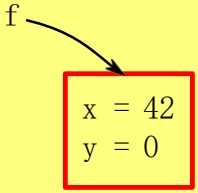
And set it to 42

f

```
x = 42
y = 0
```

A simple class — changing the object

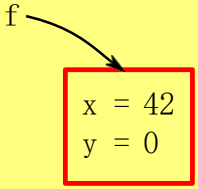
```
Foo f = new Foo();
f.x = 42;
f.y = 9;
```



A simple class — changing the object

Look in the object labelled f

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
```



A simple class — changing the object

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
```

Look in the object labelled f

Find its y variable

f

x = 42
y = 0

A simple class — changing the object

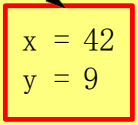
```
Foo f = new Foo();
f.x = 42;
f.y = 9;
```

Look in the object labelled f

Find its y variable

And set it to 9

f



A simple class — reading the object

```
class Foo {
  int x;
  int y;
}

Foo f = new Foo ();
f. x = 42;
f. y = 9;
System.out.println(f.x);
```

A simple class — reading the object

```
class Foo {
  int x;
  int y;
}

Foo f = new Foo ();
f. x = 42;
f. y = 9;
System.out.println(f.x);
```

What just happened?

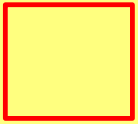
A simple class — reading the object

```
Foo f = new Foo();
```

A simple class — reading the object

```
Foo f = new Foo();
```

Create a new object...



A simple class — reading the object

...using a plan (or `class') called Foo...

```
Foo f = new Foo();
```

Create a new object...

```
x = 0
y = 0
```

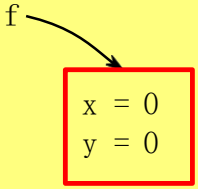
A simple class — reading the object

...using a plan (or `class') called Foo...

```
Foo f = new Foo();
```

...and give it the label f

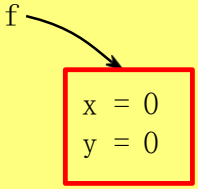
Create a new object...



A simple class — reading the object

Look in the object labelled f

```
Foo f = new Foo();
f.x = 42;
```



A simple class — reading the object

Look in the object labelled f

```
Foo f = new Foo();
f.x = 42;
```

Find its x variable

f

```
x = 0
y = 0
```

A simple class — reading the object

```
Foo f = new Foo();
f.x = 42;
```

Look in the object labelled f

Find its x variable

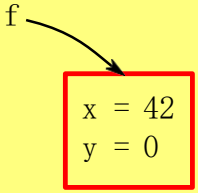
And set it to 42

f

```
x = 42
y = 0
```

A simple class — reading the object

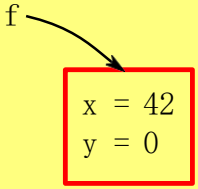
```
Foo f = new Foo();
f.x = 42;
f.y = 9;
```



A simple class — reading the object

Look in the object labelled f

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
```



A simple class — reading the object

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
```

Look in the object labelled f

Find its y variable

f

x = 42
y = 0

A simple class — reading the object

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
```

Look in the object labelled f

Find its y variable

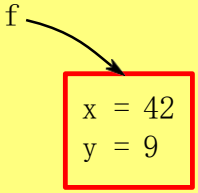
And set it to 9

f

```
x = 42
y = 9
```

A simple class — reading the object

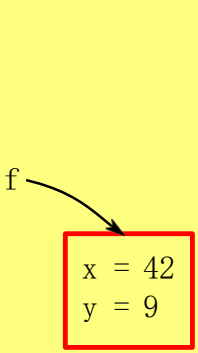
```
Foo f = new Foo();
f.x = 42;
f.y = 9;
System.out.println(f.x);
```



A simple class — reading the object

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
System.out.println(f.x);
```

Look in the object labelled f



A simple class — reading the object

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
System.out.println(f.x);
```

Look in the object labelled f

Give me its x variable

f

```
x = 42
y = 9
```

A simple class — reading the object

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
System.out.println(f.x);
```

And print it out

Look in the object labelled f

Give me its x variable

f

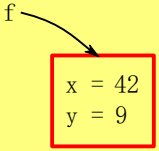
```
x = 42
y = 9
```

Exercise

Let’s try this ourselves. Look at exercise 1 on Moodle.

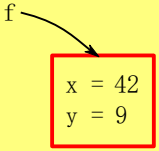
Multiple objects of the same class

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
```



Multiple objects of the same class

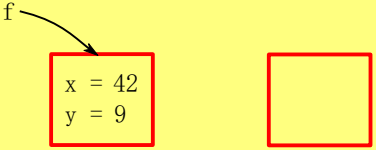
```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
```



Multiple objects of the same class

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
```

Create a new object...

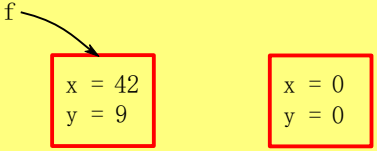


Multiple objects of the same class

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
```

...using a plan (or `class') called Foo...

Create a new object...



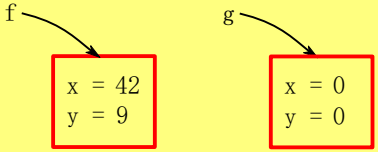
Multiple objects of the same class

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
```

...and give it the label g

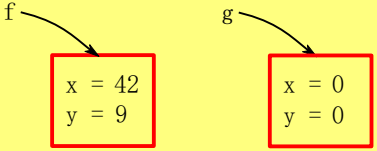
Create a new object...

...using a plan (or `class') called Foo...



Multiple objects of the same class

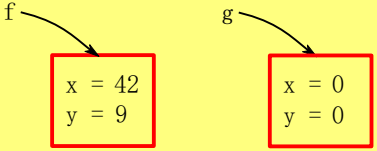
```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
g.x = 66;
```



Multiple objects of the same class

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
g.x = 66;
```

Look in the object labelled g

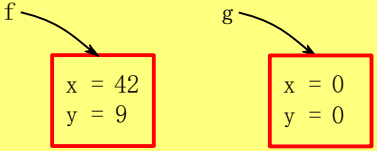


Multiple objects of the same class

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
g.x = 66;
```

Look in the object labelled g

Find its x variable



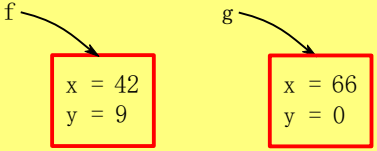
Multiple objects of the same class

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
g.x = 66;
```

Look in the object labelled g

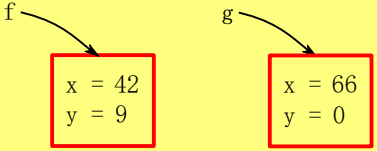
Find its x variable

And set it to 66



Multiple objects of the same class

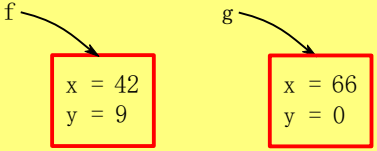
```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
g.x = 66;
g.y = 25;
```



Multiple objects of the same class

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
g.x = 66;
g.y = 25;
```

Look in the object labelled g

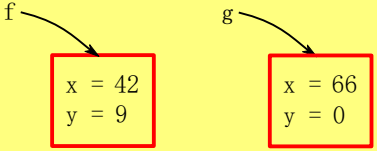


Multiple objects of the same class

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
g.x = 66;
g.y = 25;
```

Look in the object labelled g

Find its y variable



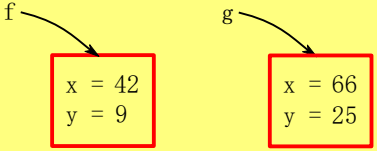
Multiple objects of the same class

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = new Foo();
g.x = 66;
g.y = 25;
```

Look in the object labelled g

Find its y variable

And set it to 25



References

- These labels are more properly called references (sometimes pointers).

References

- These labels are more properly called <u>references</u> (sometimes <u>pointers</u>).
- They refer to or point to an object.

References

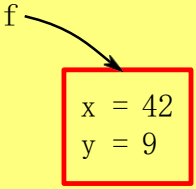
- These labels are more properly called <u>references</u> (sometimes <u>pointers</u>).
- They refer to or point to an object.
- You can have more than one reference to the same object.

References

- These labels are more properly called <u>references</u> (sometimes <u>pointers</u>).
- They refer to or point to an object.
- You can have more than one reference to the same object.
- If you have no references, you can't find the object any more (and Java will destroy it).

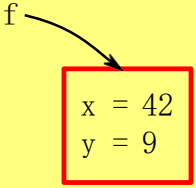
References

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
```



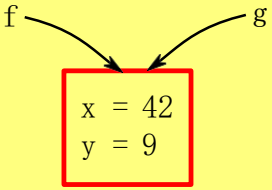
References

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = f;
```



References

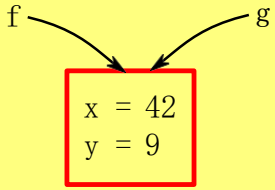
```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = f;
```



References

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = f;
g.x = 77;
```

Look in the object labelled g

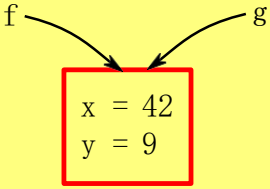


References

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = f;
g.x = 77;
```

Look in the object labelled g

Find its x variable



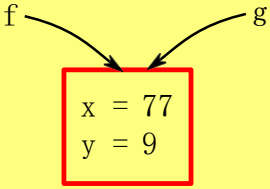
References

```
Foo f = new Foo();
f.x = 42;
f.y = 9;
Foo g = f;
g.x = 77;
```

Look in the object labelled g

Find its x variable

And set it to 77



Why?

- To collect some related variables into one place.

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
int player X = 1;
int player Y = 5;
int baddie X = 8;
int baddie Y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
class Coordinate {
  int x;
  int y;
}

Coordinate player = new Coordinate ();
player.x = 1;
player.y = 5;
Coordinate baddie = new Coordinate ();
baddie.x = 8;
baddie.y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
class Coordinate {
  int x;
  int y;
}
```

```
Coordinate player = new Coordinate ();
player.x = 1;
player.y = 5;
Coordinate baddie = new Coordinate ();
baddie.x = 8;
baddie.y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
class Coordinate {
  int x;
  int y;
}

Coordinate player = new Coordinate ();
player.x = 1;
player.y = 5;
Coordinate baddie = new Coordinate ();
baddie.x = 8;
baddie.y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
class Coordinate {
  int x;
  int y;
}

Coordinate player = new Coordinate ();
player.x = 1;
player.y = 5;
Coordinate baddie = new Coordinate ();
baddie.x = 8;
baddie.y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
class Coordinate {
  int x;
  int y;
}

Coordinate player = new Coordinate ();
player.x = 1;
player.y = 5;
Coordinate baddie = new Coordinate ();
baddie.x = 8;
baddie.y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
class Coordinate {
  int x;
  int y;
}

Coordinate player = new Coordinate ();
player .x = 1;
player.y = 5;
Coordinate baddie = new Coordinate ();
baddie.x = 8;
baddie.y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
class Coordinate {
  int x;
  int y;
}

Coordinate player = new Coordinate ();
player .x = 1;
player .y = 5;
Coordinate baddie = new Coordinate ();
baddie.x = 8;
baddie.y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
class Coordinate {
  int x;
  int y;
}

Coordinate player = new Coordinate();
player.x = 1;
player.y = 5;
Coordinate baddie = new Coordinate();
baddie.x = 8;
baddie.y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
class Coordinate {
  int x;
  int y;
}

Coordinate player = new Coordinate ();
player.x = 1;
player.y = 5;
Coordinate baddie = new Coordinate ();
baddie .x = 8;
baddie .y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. a coordinate:

```
class Coordinate {
  int x;
  int y;
}

Coordinate player = new Coordinate ();
player.x = 1;
player.y = 5;
Coordinate baddie = new Coordinate ();
baddie.x = 8;
baddie.y = 12;
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
String personAname = "Fred";
String personAaddress = "10 Downing Street";
String personAphone = "999";
String personBname = "Sheila";
String personBaddress = "1060 West Addison";
String personBphone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Why?

- To collect some related variables into one place.
- e.g. information about a person:

```
class Person {
String name;
String address;
String phone;
}

Person personA = new Person();
personA.name = "Fred";
personA.address = "10 Downing Street";
personA.phone = "999";

Person personB = new Person();
personB.name = "Sheila";
personB.address = "1060 West Addison";
personB.phone = "0714159132526";
```

Exercise

Let’s do some more coding with classes and objects. Look at exercise 2 on Moodle.

null references

What if we have:

```
class Ostrabagalous {
  int x;
  String y;
}

Ostrabagalous o;
```

null references

What if we have:

```
class Ostrabagalous {
  int x;
  String y;
}

Ostrabagalous o;
```

- What does `o` refer to?

null references

What if we have:

```
class Ostrabagalous {
  int x;
  String y;
}

Ostrabagalous o;
```

- What does `o` refer to?
- Nothing. It refes to no object.

null references

What if we have:

```
class Ostrabagalous {
  int x;
  String y;
}

Ostrabagalous o;
```

- What does `o` refer to?
- Nothing. It refes to no object.
- This is called `null` in Java.

null references

What if we have:

```
class Ostrabagalous {
  int x;
  String y;
}

Ostrabagalous o;
```

- What does `o` refer to?
- Nothing. It refes to no object.
- This is called `null` in Java.
- What happens if I do

  ```
  o.x = 42;
  ```

References with methods

What happens if I do this:

```
void main(String[] args) {
  int x = 42;

  myGreatMethod(x);
}

void myGreatMethod(int n) {
  System.out.println(n);
}
```

References with methods

```
void main(String[] args) {
  int x = 42;
  myGreatMethod(x);
}

void myGreatMethod(int n) {
  System.out.println(n);
}
```

References with methods

```
void main(String[] args) {
  int x = 42;
  myGreatMethod(x);
}

void myGreatMethod(int n) {
  System.out.println(n);
}
```

References with methods

```
void main(String[] args) {
   int x = 42;
   myGreatMethod(x);
}

void myGreatMethod(int n) {
   System.out.println(n);
}
```

x	42

Call `myGreatMethod` giving it the value of `x`

References with methods

```
void main(String[] args) {
  int x = 42;
  myGreatMethod(x);
}

void myGreatMethod(int n) {
  System.out.println(n);
}
```

x	42

References with methods

```
void main(String[] args) {
  int x = 42;
  myGreatMethod(x);
}

void myGreatMethod(int n) {
  System.out.println(n);
}
```

x	42

Call `myGreatMethod` giving it the value of `x`

References with methods

```
void main(String[] args) {
  int x = 42;
  myGreatMethod(x);
}

void myGreatMethod(int n) {
  System.out.println(n);
}
```

x	42

References with methods

```
void main(String[] args) {
  int x = 42;
  myGreatMethod(x);
}

void myGreatMethod(int n) {
  System.out.println(n);
}
```

x	42

Copy the value of x into a new variable called n

References with methods

```
void main(String[] args) {
   int x = 42;
   myGreatMethod(x);
}

void myGreatMethod(int n) {
   System.out.println(n);
}
```

x	42

References with methods

```
void main(String[] args) {
   int x = 42;
   myGreatMethod(x);
}

void myGreatMethod(int n) {
   System.out.println(n);
}
```

x	42

→ Output the value of n

References with methods

It's the same with references:

```
class Frobozz {
  String a;
  String b;
}

void main ( String [] args ) {
  Frobozz f = new Frobozz();
  f.a = "Hello world";
  myGreatMethod(f);
}

void myGreatMethod(Frobozz g) {
  System.out.println(g.a);
}
```

References with methods

It's the same with references:

```
class Frobozz {
  String a;
  String b;
}

void main ( String [] args ) {
  Frobozz f = new Frobozz();
  f.a = "Hello world";
  myGreatMethod(f);
}

void myGreatMethod(Frobozz g) {
  System.out.println(g.a);
}
```

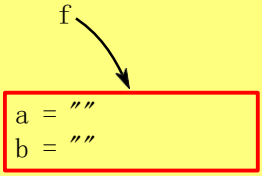
References with methods

It's the same with references:

```
class Frobozz {
   String a;
   String b;
}

void main ( String [] args ) {
   Frobozz f = new Frobozz();
   f.a = "Hello world";
   myGreatMethod(f);
}

void myGreatMethod(Frobozz g) {
   System.out.println(g.a);
}
```



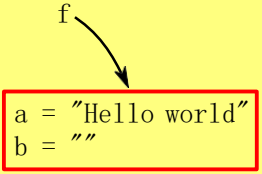
References with methods

It's the same with references:

```
class Frobozz {
  String a;
  String b;
}

void main ( String [] args ) {
  Frobozz f = new Frobozz();
  f.a = "Hello world";
  myGreatMethod(f);
}

void myGreatMethod(Frobozz g) {
  System.out.println(g.a);
}
```



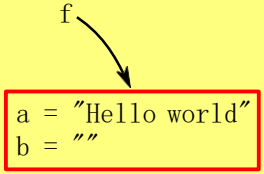
References with methods

It's the same with references:

```
class Frobozz {
  String a;
  String b;
}

void main ( String [] args ) {
  Frobozz f = new Frobozz();
  f.a = "Hello world";
  myGreatMethod(f);
}

void myGreatMethod(Frobozz g) {
  System.out.println(g.a);
}
```



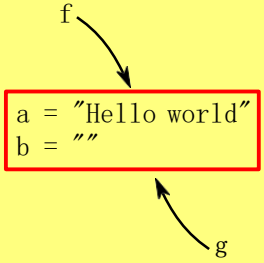
References with methods

It's the same with references:

```
class Frobozz {
   String a;
   String b;
}

void main ( String [] args ) {
   Frobozz f = new Frobozz();
   f.a = "Hello world";
   myGreatMethod(f);
}

void myGreatMethod(Frobozz g) {
   System.out.println(g.a);
}
```



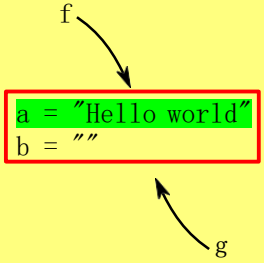
References with methods

It's the same with references:

```
class Frobozz {
   String a;
   String b;
}

void main ( String [] args ) {
   Frobozz f = new Frobozz();
   f.a = "Hello world";
   myGreatMethod(f);
}

void myGreatMethod(Frobozz g) {
   System.out.println(g.a);
}
```



Exercise

Look at exercise 3 on Moodle.

Summary

- Today we looked at classes and references.

Summary

- Today we looked at classes and references.
- These concepts are <u>hard</u> and <u>important</u>.

Summary

- Today we looked at classes and references.
- These concepts are hard and important.
- Make sure you understand them!

Summary

- Today we looked at classes and references.
- These concepts are hard and important.
- Make sure you understand them!
- Ask for help if you need it!

Summary

- Today we looked at classes and references.
- These concepts are <u>hard</u> and <u>important</u>.
- Make sure you understand them!
- Ask for help if you need it!
- The rest of it is easy(-ish) if you really 'get' these ideas.

Summary

- Today we looked at classes and references.
- These concepts are <u>hard</u> and <u>important</u>.
- Make sure you understand them!
- Ask for help if you need it!
- The rest of it is easy(-ish) if you really 'get' these ideas.
- Finish all the exercises from the lecture before next week.