technocamps

UNDEB EWROPEAIDD
EUROPEAN UNION

Llywodraeth Cymru
Welsh Government

Cronfa Gymdeithasol Ewrop
European Social Fund

Prifysgol Abertawe
Swansea University

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

PRIFYSGOL BANGOR UNIVERSITY

Cardiff Metropolitan University | Prifysgol Metropolitan Caerdydd

i.t.wales

PRIFYSGOL ABERYSTWYTH UNIVERSITY

Prifysgol Glyndŵr Wrecsam | Wrexham Glyndŵr University

University of South Wales
Prifysgol De Cymru

Greenfoot

Computers vs. Humans

Discuss for 30 seconds and come up with an answer. Yes or no? Why?

technocamps

Computers vs. Humans

Discuss for 30 seconds and come up with an answer. Yes or no? Why?

They are certainly quicker, and they can't make mistakes but there is nothing a computer can do that humans can't.

All we need to do to solve problems like a computer, is to think like a computer.

This is called Computational Thinking.

Activity: Follow These Instructions

You'll need a pen and paper for this task. You are going to be given instructions which you'll need to complete.

You are not allowed to ask any questions or discuss with anyone!

Activity: Follow These Instructions

You'll need a pen and paper for this task. You are going to be given instructions which you'll need to complete.

You are not allowed to ask any questions or discuss with anyone!

You've all drawn perfectly good houses, right?

What went wrong? Whose fault is it? Is it your fault? Is it your teacher's fault? Is it my fault?

Why?

technocamps

Would This Work Better?

Year 6 pupil's suggestion:

"Just tell everyone to draw a house!"

So what would a computer have done?

technocamps

A Good Quote

"The computer is incredibly fast, accurate, and stupid. Man is incredibly slow, inaccurate, and brilliant. The marriage of the two is a force beyond calculation." – Leo Cherne

Computers are everywhere but people do not understand how to use them for everyday tasks.

My recent examples:

- Simple python program/Excel formula for adding ";" to the end of Email addresses
- Booklet arranger which calculates how to rearrange pages for making A5 style-booklets.

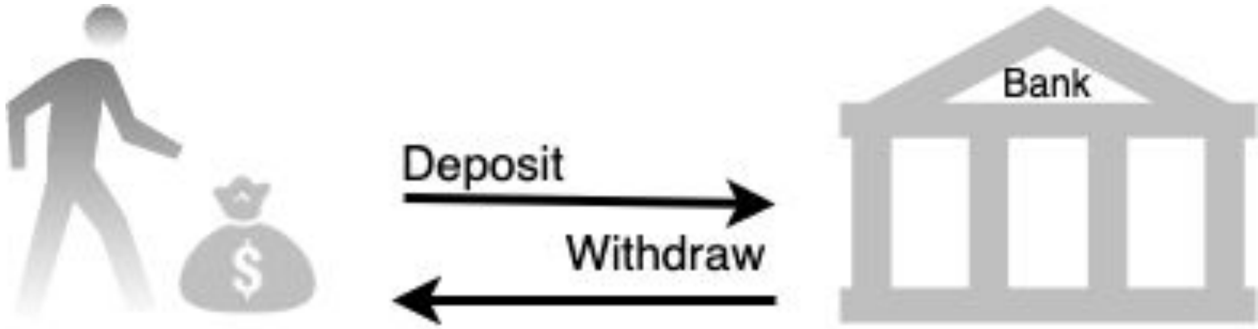
Programming Paradigms

Programming Paradigm is a style or way of programming.

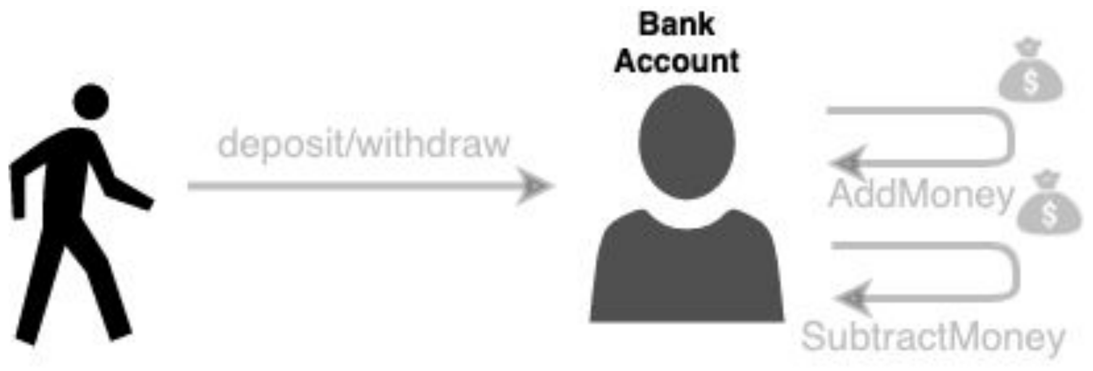
There are different ways or styles of programming.

- Procedural Programming: Python
- Event-driven Programming: Scratch
- Object-Oriented Programming: Java

Procedural Programming



Object-Oriented Programming



Sequential vs. OOP

Object-oriented programming is a way of programming which is slightly different to Python.

It is structured differently to Python’s sequential way of coding.

Java models solutions based on real world scenario. It uses Classes and Objects.

Does anyone know what these are? Has anyone used them before?

Why Java?

Java as a programming language is in high demand throughout the business industries.

Object-oriented programming model is widely used in the industry, be it the cloud, server or embedded industry.

Junior Java Software Engineer
Priocept - London
£18,000 - £35,000

Java Developer
Thecitysecret Ltd - Richmond
£40,000 - £50,000

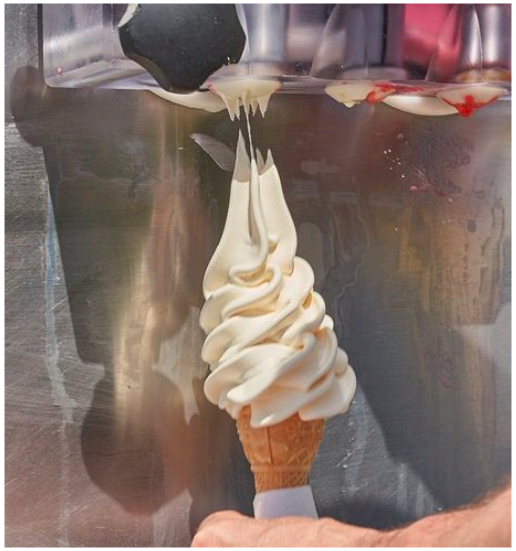
Java Developer
Rated People - London
£50,000 - £60,000

Java Developer
Acorn Recruitment - Swansea
£35,000 - £50,000

Ice cream!!!

Class:

A sweetened frozen food, usually made from milk, typically eaten as a dessert.



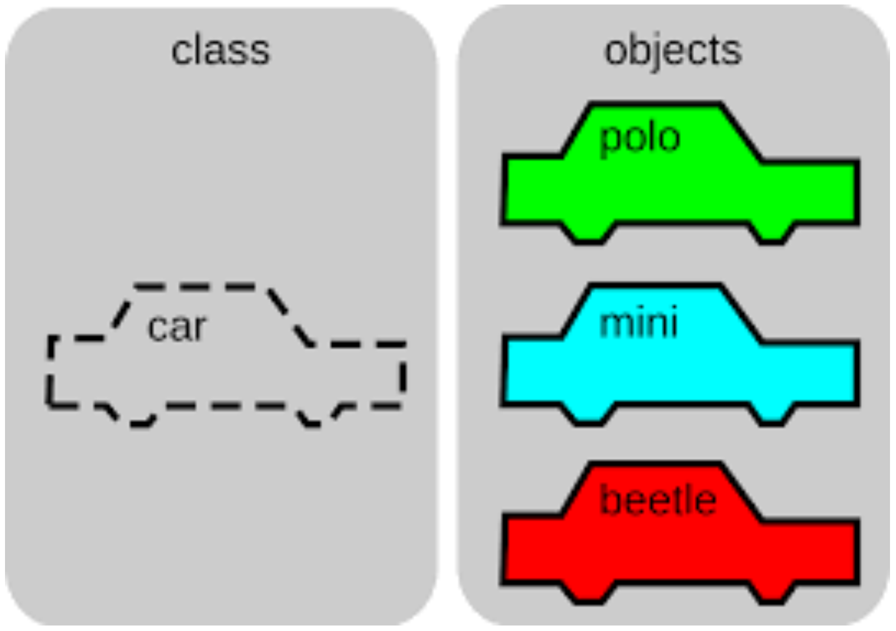
Objects:

Vanilla Ice Cream
Chocolate Ice Cream
Mint Choc Chip Ice Cream
Salted Caramel Ice Cream

Class and Objects

A **Class** describes the contents of the objects that belong to it: it describes the properties and the operations/behaviour.

An **Object** is an element of a class; objects have the behaviours of their class. The object is the actual component.



Class: Student

Imagine I had a class called Student, what properties does a student have i.e. what makes a student a student?

technocamps

Class: Student

Imagine I had a class called Student, what properties does a student have i.e. what makes a student a student?

Student:

Name
Age
Grade

Class: Student

Apart from the properties, what can a student **do**?

Student:

Name
Age
Grade

Class: Student

Apart from the properties, what can a student **do**?

Student:

Name
Age
Grade
doHomework()
play()

technocamps

Student Class and Objects

Tom
11
7

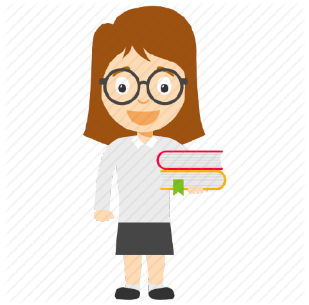
(Class)Student:

Name
Age
Grade
doHomework()
play()

technocamps

Student Class and Objects

Tom
11
7



Casey
13
9

(Class)Student:

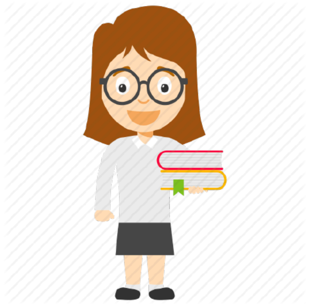
Name
Age
Grade
doHomework()
play()

Student Class and Objects

Tom
11
7

(Class)Student:

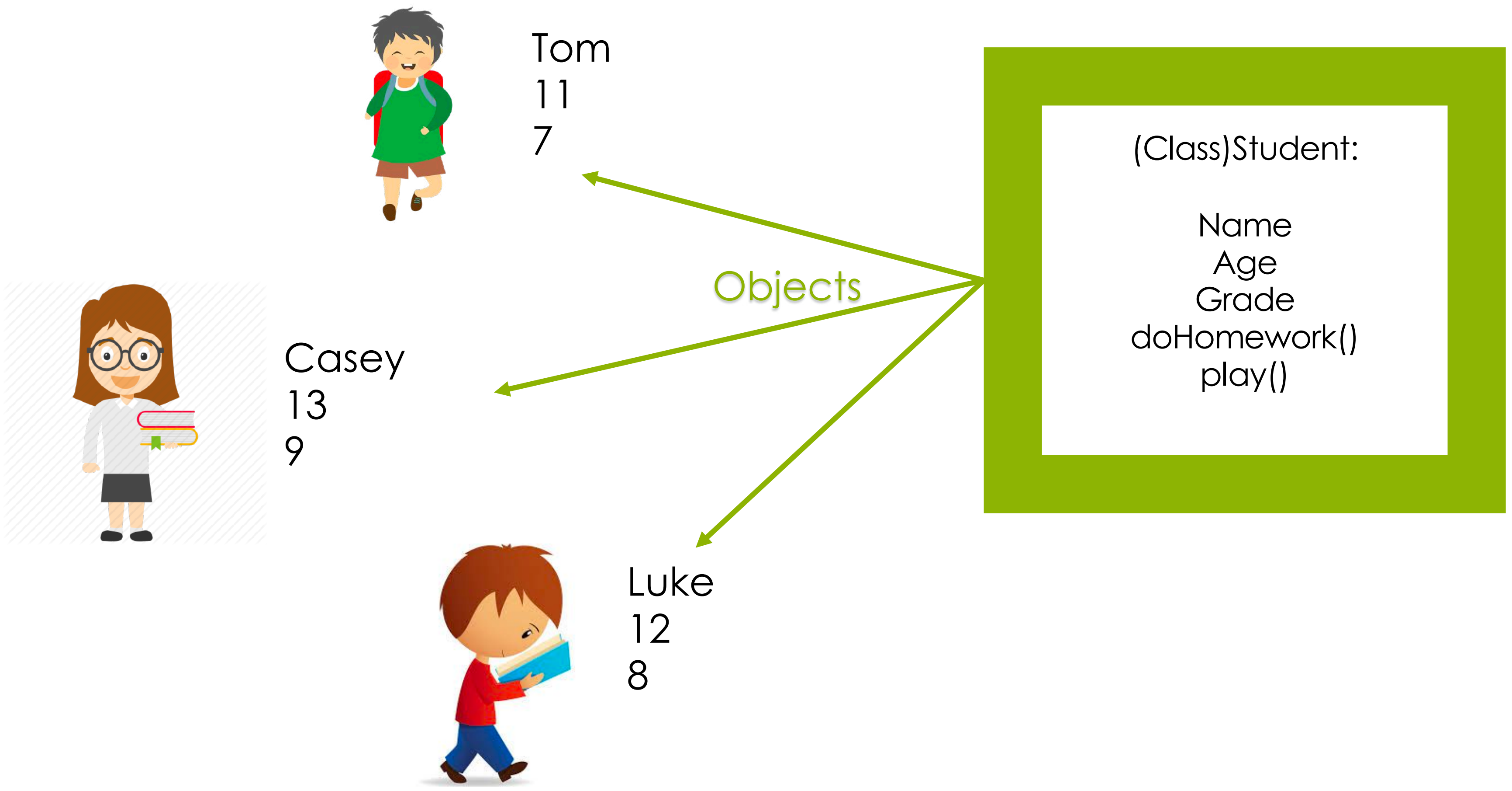
Name
Age
Grade
doHomework()
play()



Casey
13
9

Luke
12
8

Student Class and Objects



Activity: Identify Class

What Class do these Objects belong to?

Apple Mac Pro

Microsoft Surface Pro

Dell Inspirion 15

Lenovo Yoga

Activity: Identify Class

What Class do these Objects belong to?

Laptops

Apple Mac Pro

Microsoft Surface Pro

Dell Inspirion 15

Lenovo Yoga

Activity: Identify Objects

Identify the Objects that belong to the Class **MobilePhones**

Motorola G7

PS4

Apple iPhone X

Canon 70D

Huawei P30 Pro

Samsung Galaxy S5

X-Box

technocamps

Activity: Identify Objects

Identify the Objects that belong to the Class **MobilePhones**

Motorola G7

~~PS4~~

Apple iPhone X

~~Canon 70D~~

Huawei P30 Pro

Samsung Galaxy S5

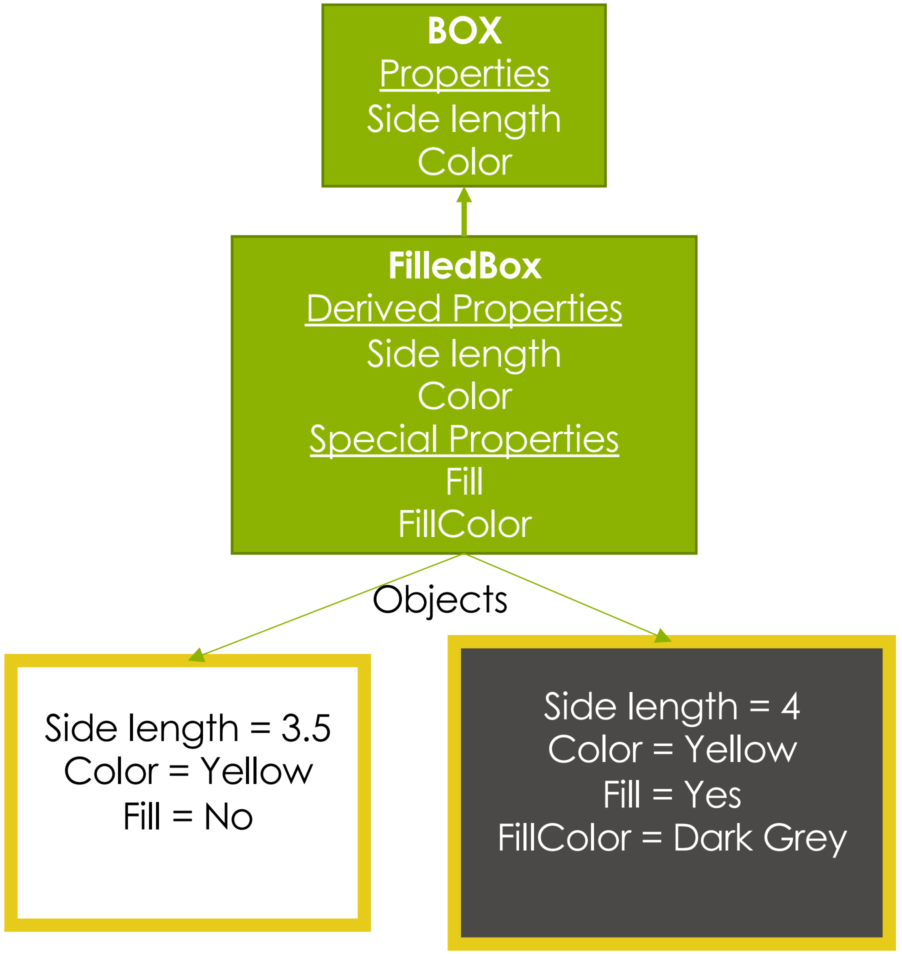
~~X-Box~~

Inheritance

What do you think inheritance means in terms of classes?

Inheritance

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviours of a parent class. It is an important part of an Object-Oriented programming system.

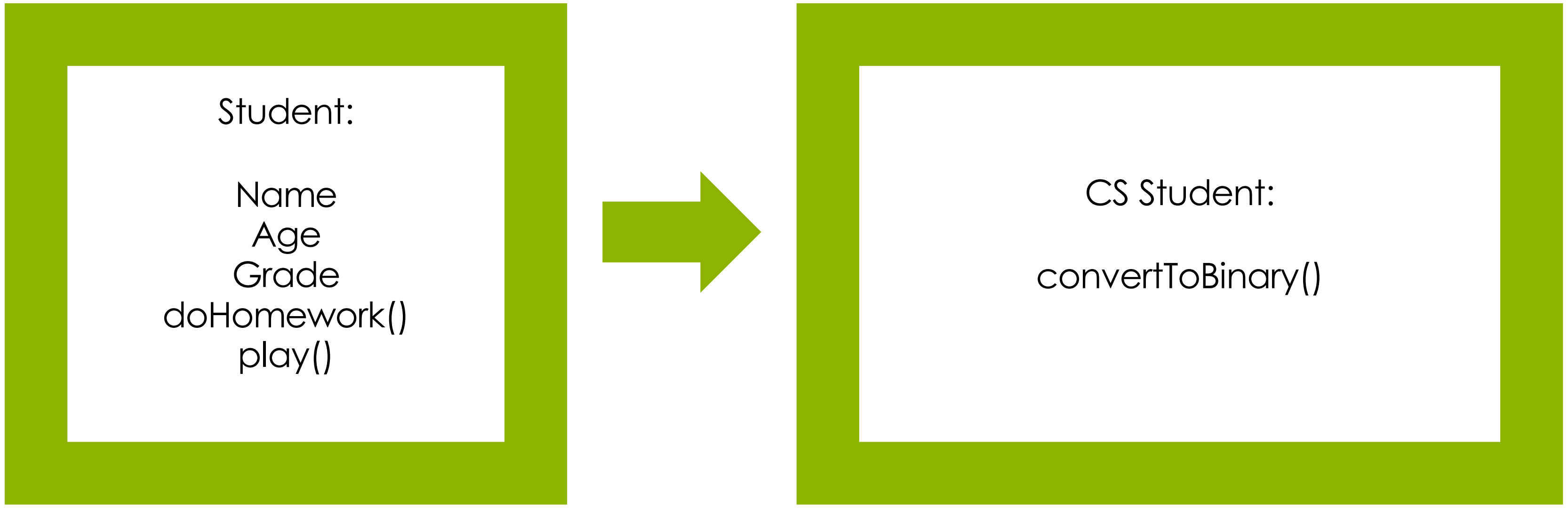


Inheritance

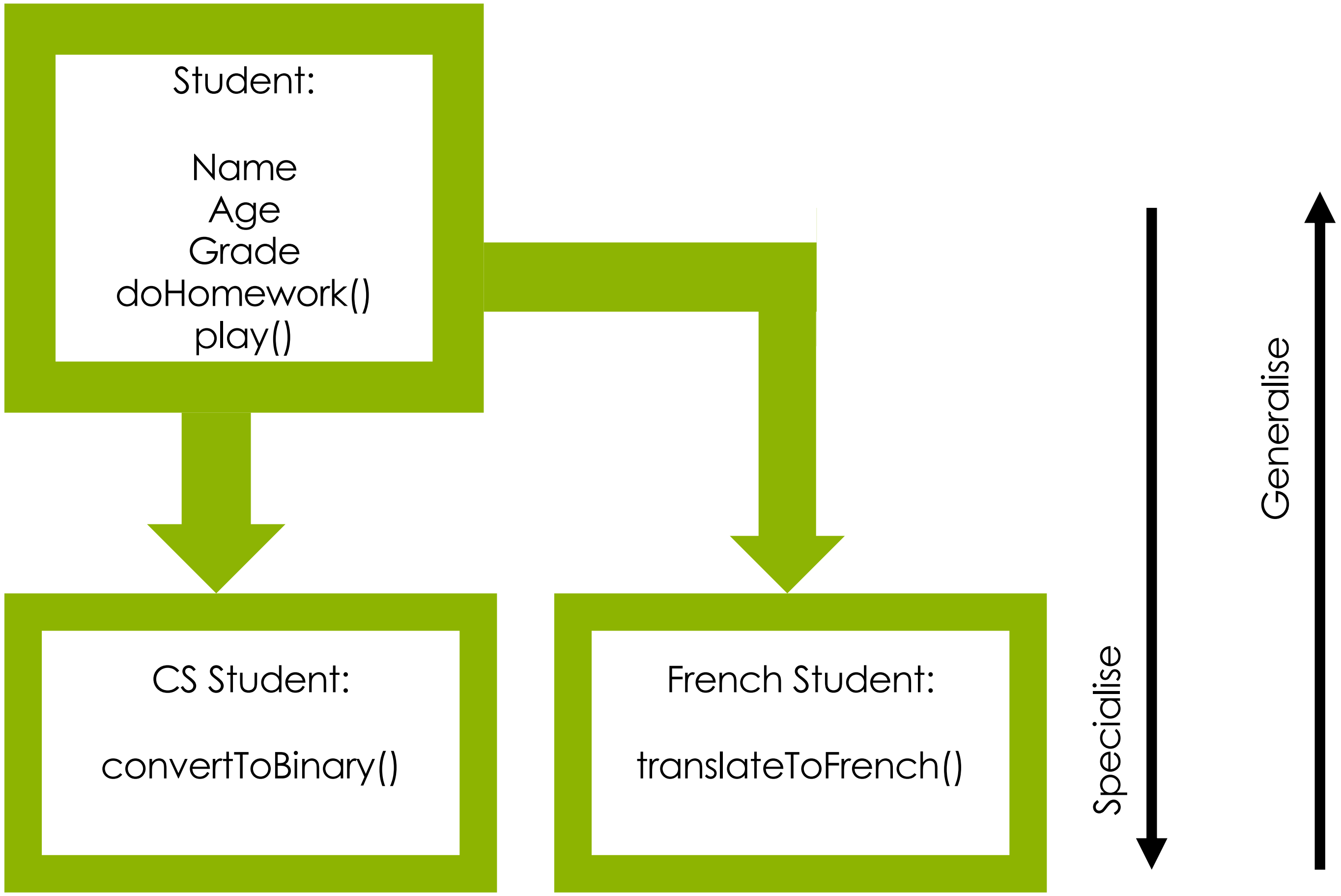
Student:

Name
Age
Grade
doHomework()
play()

Inheritance



Inheritance

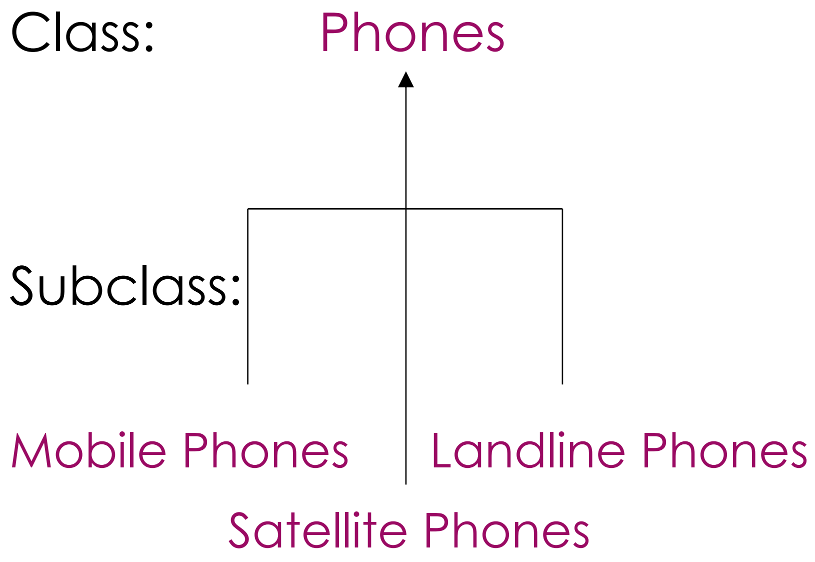


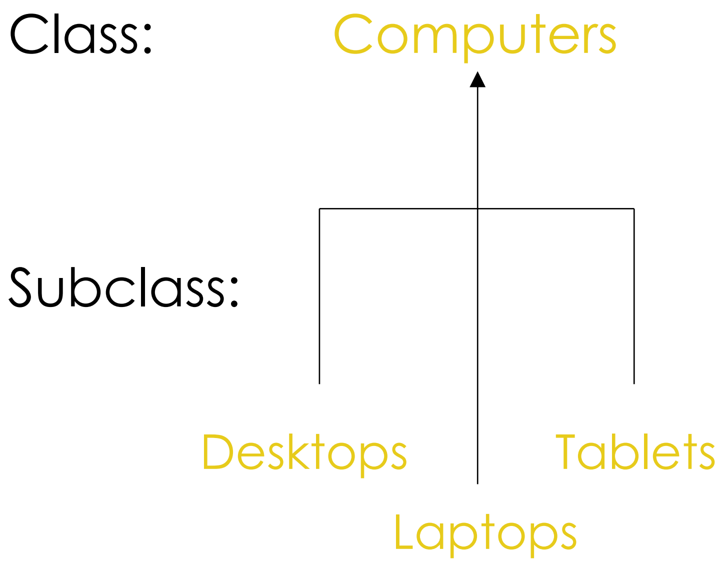
technocamps

Activity: Identify Class and Subclasses

Set 1:	Set 2:
Mobile Phones	Tablets
Satellite Phones	Laptops
Phones	Desktops
Landline Phones	Computers

Activity: Identify Class and Subclasses





What is Greenfoot?

Greenfoot is an introductory visual programming environment using the Java programming language.

It is a great language to learn and incorporates a textual experience of programming, providing users with great initial programming knowledge and understanding of basic principles of Object-oriented programming.

One of the benefits of using Greenfoot for an introduction into programming is not only the helpful guides and tutorials available, but also the colourful user friendly interface.

http://www.greenfoot.org/download

Greenfoot Version 2.4.2

The version we will be using is Greenfoot Version 2.4.2 in order to make sure we're all on the same page and have the same methods/functions available to us.

World and Actor

All the world's a stage,
And all the men and women merely players;

- As you like it, William Shakespeare

Greenfoot has two main classes:

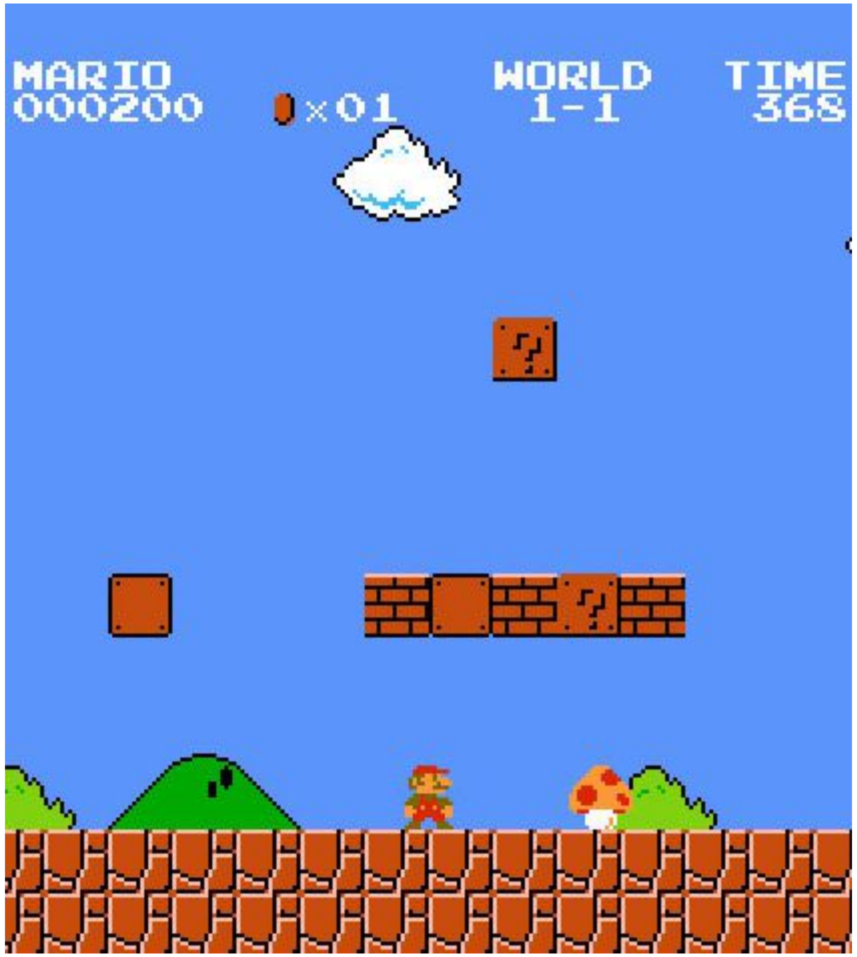
a) **World**: A world where all actors live and interact. Every instance of the World is different, e.g. Jungle, Space, Sea etc.

b) **Actor**: An object of the Actor exists in the World. Each actor has its own characteristics and behavior. E.g. Predator and Prey objects in a Jungle, Aliens and Astronomer objects in Space, and Fish and Shark objects in Sea.

A **Scenario** in Greenfoot is equal to Object(s) of Actor Class + World Class + Programming rules.

technocamps

Activity: Actors and World



If we look at Super Mario for example, which parts are the Actors and which parts are the World?

Discuss with a partner which objects are part of the Actor class and which objects are part of the World class.

technocamps

Greenfoot Coordinate System

X

Y

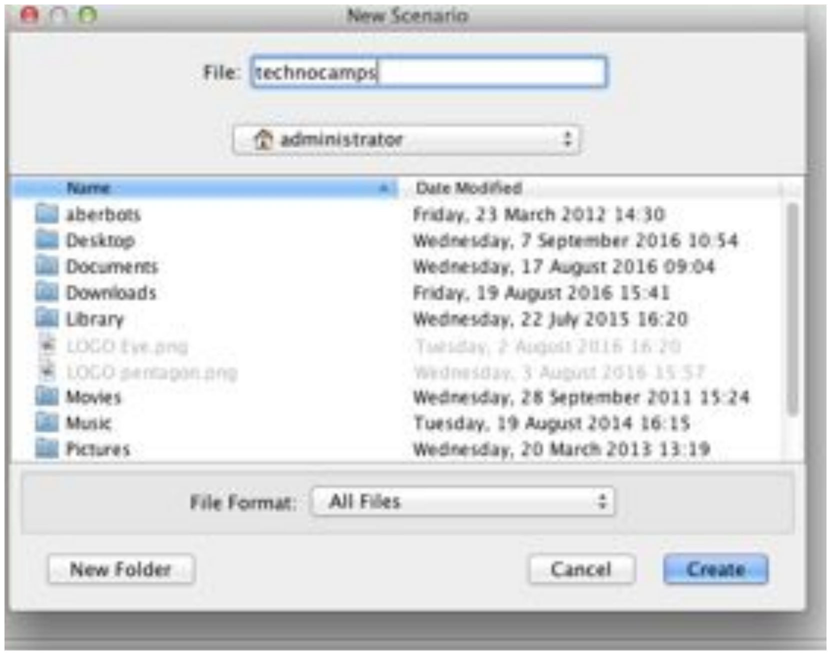
X,Y	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0,0	1,0	2,0	..	..	..	..	7,0
1	0,1							..
2	0,2							..
3	..							..
4	..							..
5	..							..
6	..							..
7	0,7	..	..	..	..	..	..	7,7

technocamps

Activity: Starting Greenfoot

First create a scenario:

1. Click on ‘Scenario’ on the top left and then ‘New Scenario’.
2. Name Your Program. Do not call your Scenario “Greenfoot”!



New Scenario

File: technocamps

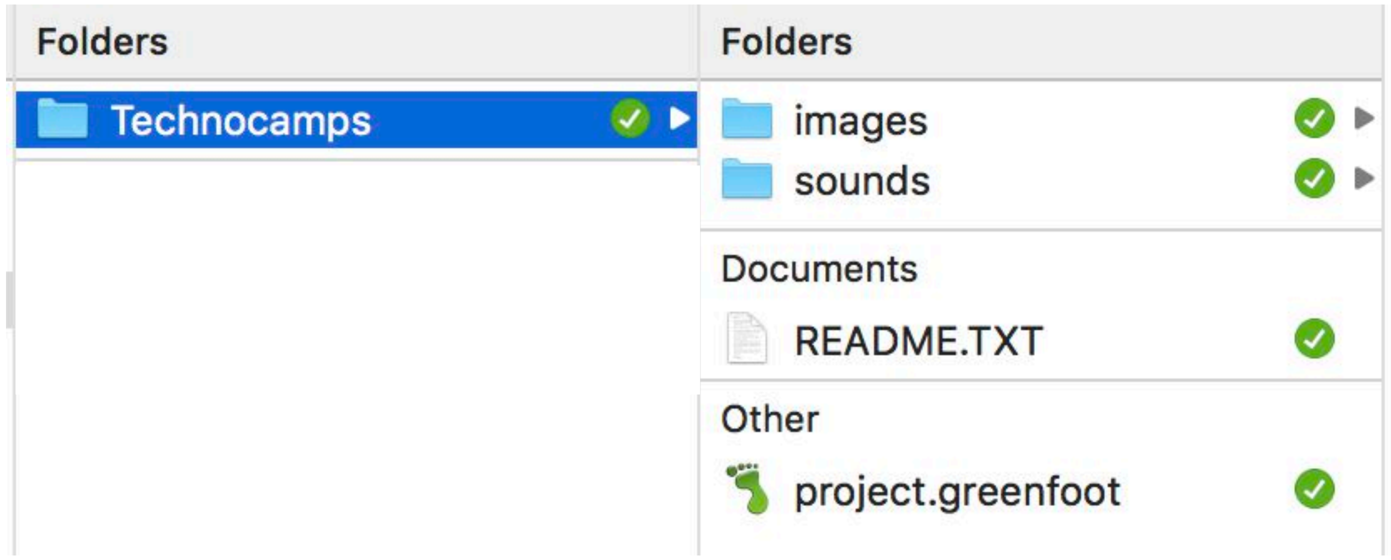
administrator

Name	Date Modified
aberbots	Friday, 23 March 2012 14:30
Desktop	Wednesday, 7 September 2016 10:54
Documents	Wednesday, 17 August 2016 09:04
Downloads	Friday, 19 August 2016 15:41
Library	Wednesday, 22 July 2015 16:20
LOGO Eye.png	Tuesday, 2 August 2016 16:20
LOGO pentagon.png	Wednesday, 3 August 2016 15:57
Movies	Wednesday, 28 September 2011 15:24
Music	Tuesday, 19 August 2014 16:15
Pictures	Wednesday, 20 March 2013 13:19

File Format: All Files

Activity: Starting Greenfoot

This creates a folder containing the project file and anything we create will also be saved automatically in this folder.



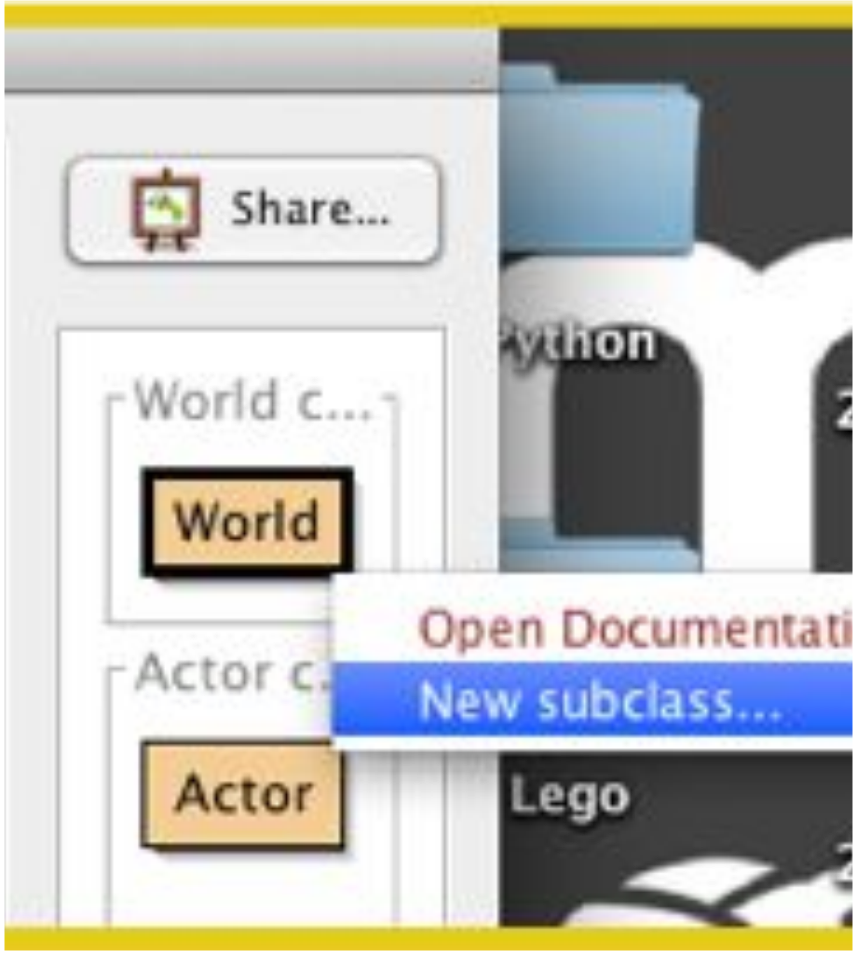
technocamps

Activity: Creating a New World

When the Greenfoot window opens it should look like the image shown.

To create our world:

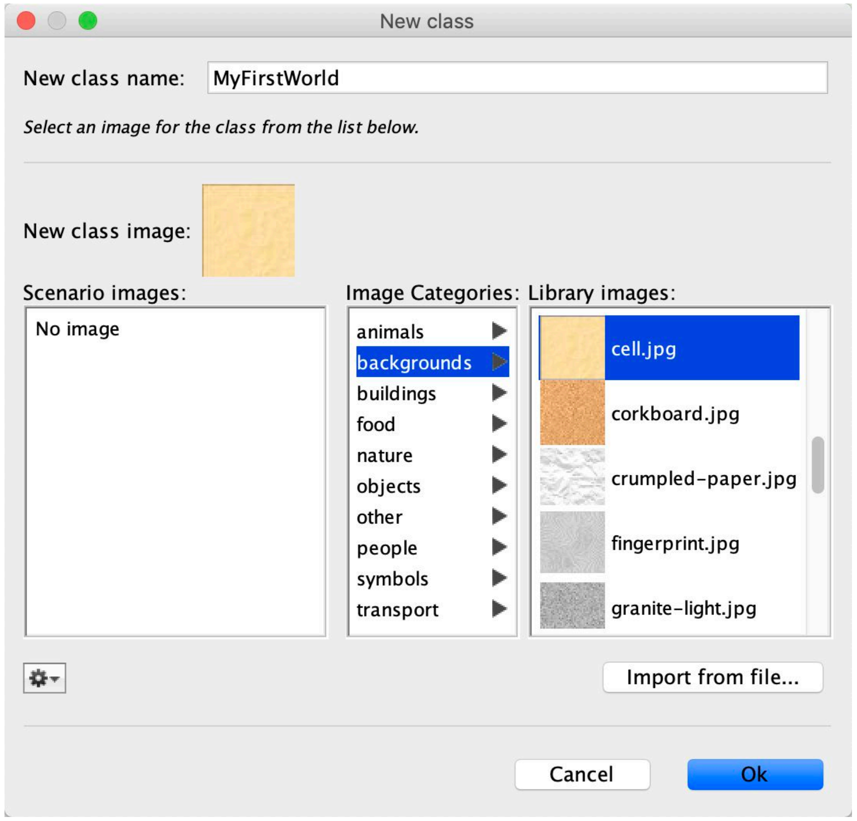
1. Right-click on 'World'
2. Select the option 'New subclass' and name that subclass. (We always start with a capital letter and capitalise the first letter of each new word, never with spaces)



technocamps

Activity: Creating a New World

3. Choose the ‘Cell’ image


New class

New class name: MyFirstWorld

Select an image for the class from the list below.

New class image:

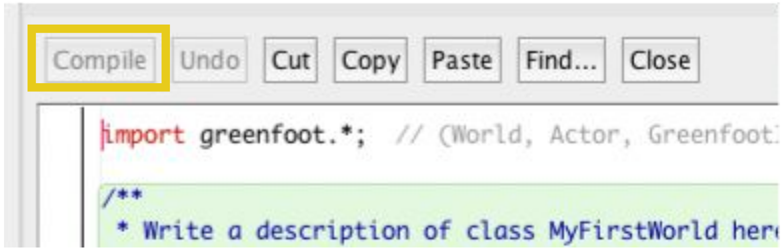
technocamps

Activity: Changing Your World

Now change your 'World' by changing the number of 'cells' or 'grids' in our program. To do this:

1. Right-click on your 'World' subclass i.e. 'MyFirstWorld'
2. Click 'Open editor'

Note: Whenever you change the code in your subclass, do not forget to click on the 'compile' button to update the changes.



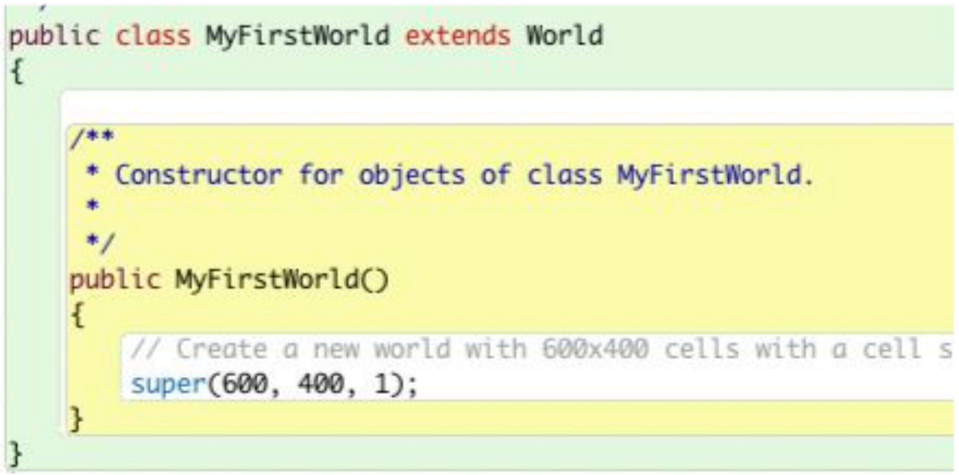
```
import greenfoot.*;  // (World, Actor, Greenfoot

/**
 * Write a description of class MyFirstWorld her
```

technocamps

Activity: Changing the Grid

Currently as default the 'World' contains a 'grid' of 600x400 with a size of 1x1 (which translates to a lot of cells, but with a very small size).

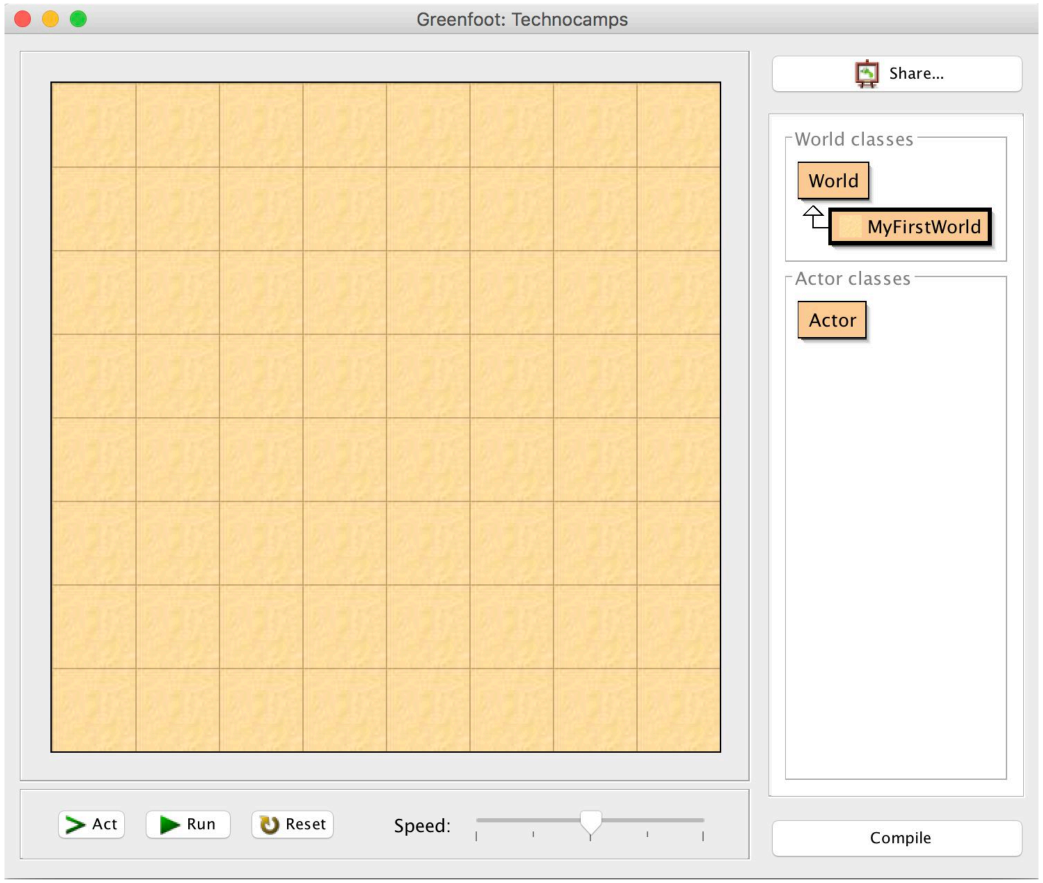
Change it so that it is 8x8 cells, that is to say it should contain 8 rows and 8 columns, and each cell has the size of 60x60 pixels.



```
public class MyFirstWorld extends World
{

    /**
     * Constructor for objects of class MyFirstWorld.
     *
     */
    public MyFirstWorld()
    {
        // Create a new world with 600x400 cells with a cell s
        super(600, 400, 1);
    }
}
```

technocamps

Your Program Should Look like This:



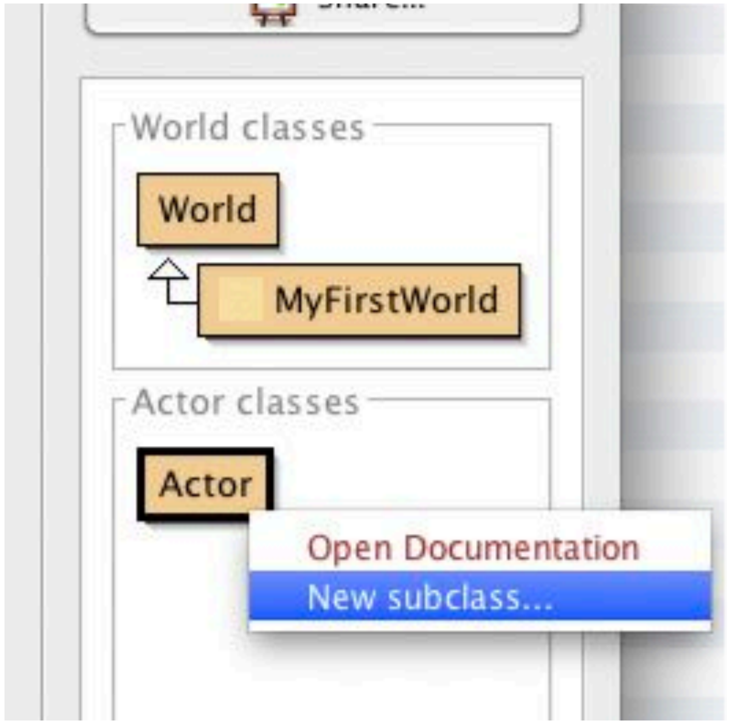
Actors

Objects of the class 'Actor' can be placed into your 'World'. Objects like:

- Main characters: hero, man, woman...
- Animals: rabbit, wombat...
- Collectables: flower, ball...

To start let us create a subclass:

1. Right-click on the 'Actor'
2. Select the option 'New subclass'



Adding Our Actor (MainCharacter)

To add our actor to the world we will need to program inside our ‘World’ subclass.

We can use the documentation to help us.

Greenfoot Documentation

Documentation is what we call the material that provides official information or evidence about the language i.e. Greenfoot.

This documentation also covers how you use different methods and what kind of inputs go inside them. These methods are things like:

'move()'

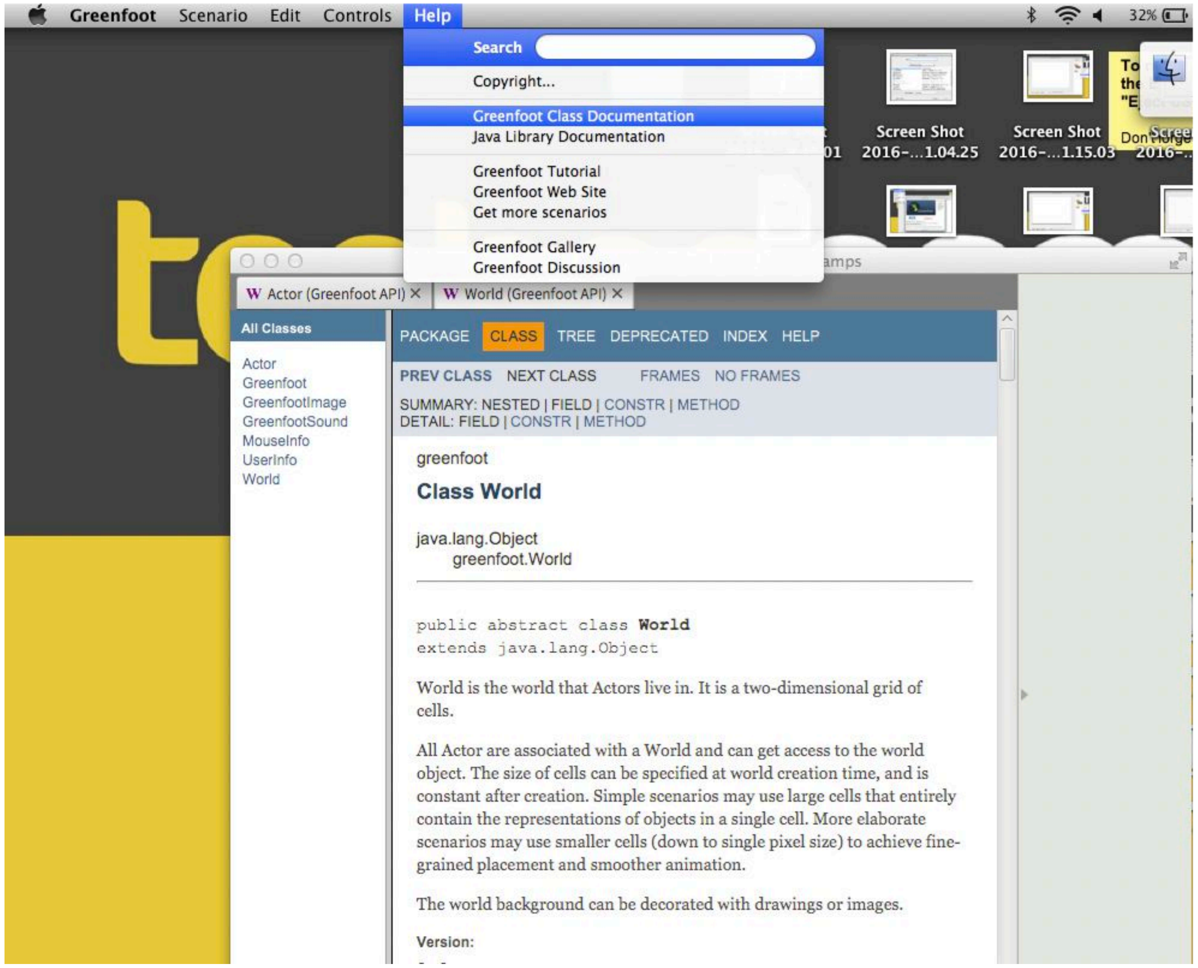
'turn()'

'removeTouching()'

'setRotation()'

The Greenfoot Class Documentation can be found as shown on the next slide.

To Find the Documentation



greenfoot

Class World

java.lang.Object
greenfoot.World

```
public abstract class World
extends java.lang.Object
```

World is the world that Actors live in. It is a two-dimensional grid of cells.

All Actor are associated with a World and can get access to the world object. The size of cells can be specified at world creation time, and is constant after creation. Simple scenarios may use large cells that entirely contain the representations of objects in a single cell. More elaborate scenarios may use smaller cells (down to single pixel size) to achieve fine-grained placement and smoother animation.

The world background can be decorated with drawings or images.

Version:

Activity: Creating Our ‘Actor’ (MainCharacter)

To do this we simply write in our World subclass:

```
MainCharacter frog = new MainCharacter();
```

The above line of code means that we are making a new object called ‘frog’. This ‘frog’ is a **new** MainCharacter which is why we use the equals symbol.

Important Note: For naming Objects, we use camelCase notation i.e.

‘frog’ or ‘thisIsMyObject’

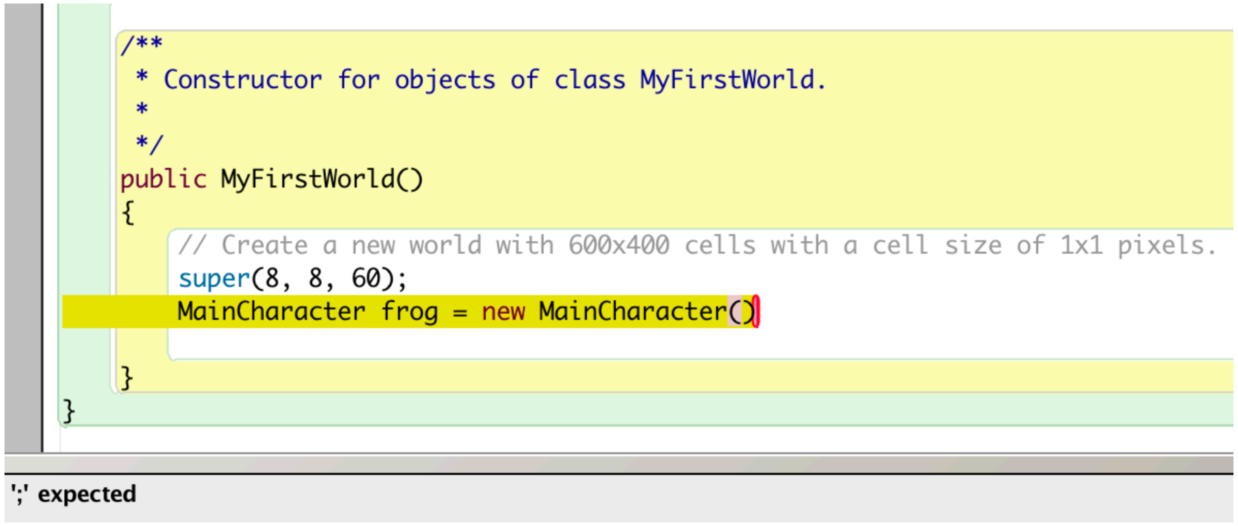
Java Syntax

Syntax is what we call the structure of statements in a language.

In Java we have to finish most lines of code with a semi colon ;

Without this the file cannot be compiled (translated for the computer to understand.)

Greenfoot rather helpfully highlights any syntax errors in your code.

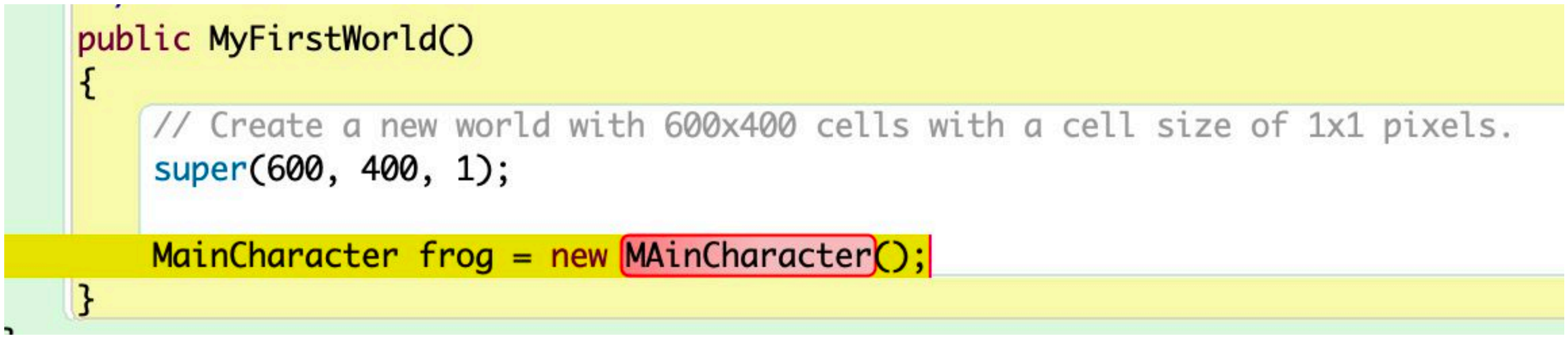


```
    /**
     * Constructor for objects of class MyFirstWorld.
     *
     */
    public MyFirstWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(8, 8, 60);
        MainCharacter frog = new MainCharacter()

    }
}
```

';' expected

Another Possible Syntax Error

Can you work out what this error means and why it happened?



```
public MyFirstWorld()
{
    // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
    super(600, 400, 1);

    MainCharacter frog = new MAinCharacter();
}
```

Java Syntax

{ }	Braces (or Curly Braces) are very important in Java. They are used to group statements and declarations.
[ ]	Brackets (or Square Brackets) are used for indexing a list.
( )	Parentheses are used to control the order of operations in an expression and to supply parameters to a method or function.

technocamps

Java Syntax

//	Double slash is used for single line comments in Java.
/*	
*	Multiple line comments in Java.
*/	
;	Every statement in Java ends with a semi-colon.

Naming Conventions

Class:	Class names should always start with **uppercase.** E.g. MainCharacter
Object:	Object names should always start with **lowercase.** E.g frogConsumer
Variable:	Variable names should start with **lowercase** and should then use **camel case.** E.g. scoreCounter
Constant:	Constants should be named using **upper case.** E.g. MAX_LIFE
Filename:	Java/Scenario file names should start with **uppercase** and continue with **camel case.** E.g. GameOfLife.java

technocamps

Activity: Adding Our ‘Actor’ (MainCharacter)

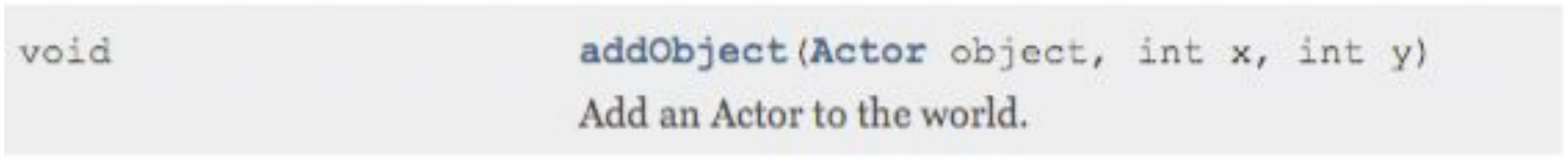
To add our frog to the world, use the Greenfoot method ‘addObject()’.

This method is already pre-built into the Greenfoot program which can be found in the ’World’ category of the Greenfoot Class Documentation website.

To open up the ‘World’ subclass (MyFirstWorld) program window, all we have to do is right-click the subclass (MyFirstWorld) and select ‘Open Editor’.

Activity: addObject()

We can find out how to use this function in the Documentation.



```
void                addObject(Actor object, int x, int y)
```

Add an Actor to the world.

As you can see, it requires 3 arguments (values) inside the brackets, what do you think these are?

Try adding the frog to your World.

Activity: Changing The Frog’s Position

It is unlikely your frog would have appeared in the very top left square of the grid.

What values would you have to put into the addObject() function to achieve this?

What about the bottom right corner?

And the other two corners?

Changing The Frog’s Behaviour

Methods are functions specific to the class, for example a class Dog may include methods such as bark() and fetch().

Other Animal subclasses may not have the same methods, i.e. a Fish class would not have the same methods as they do not behave in the same way (not even a Dogfish!).

In order to change the frog’s behaviour we need to program in the ‘Actor’ subclass and inside the ‘public void act()’ method.

If – Else Statements

If statements are used in programming in order to make decisions. If something is true a piece of code is allowed to run, if it is not true then it will not run.

General example:

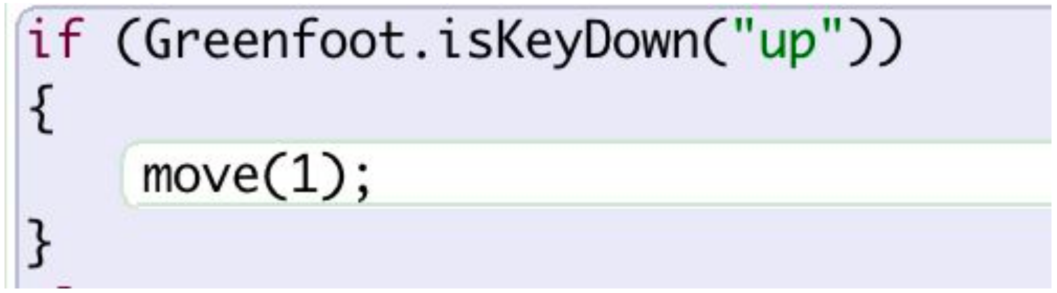
If you are wearing a jumper, raise your hand.

We can extend this to include an 'Else' as well:

If you are wearing a jumper, raise your hand. Else, stand up.

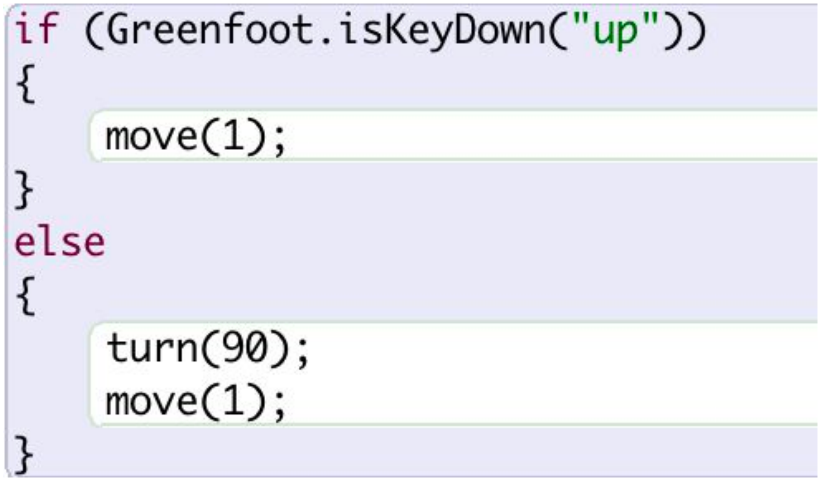
Greenfoot-Specific Example

What do you think this Greenfoot code does?



```
if (Greenfoot.isKeyDown("up"))
{
    move(1);
}
```

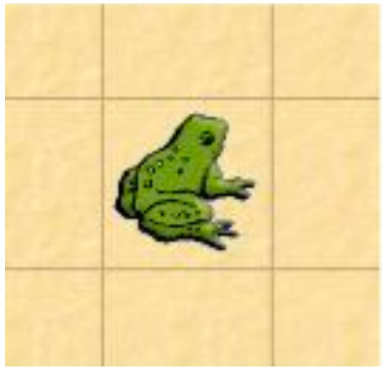
What about this?



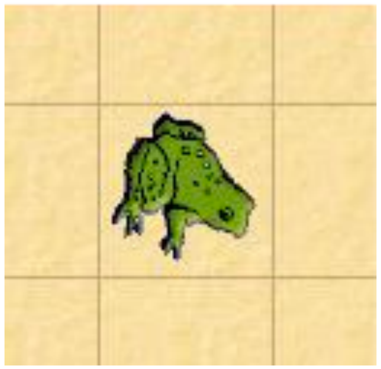
```
if (Greenfoot.isKeyDown("up"))
{
    move(1);
}
else
{
    turn(90);
    move(1);
}
```

Controlling the Frog with the Arrow Keys

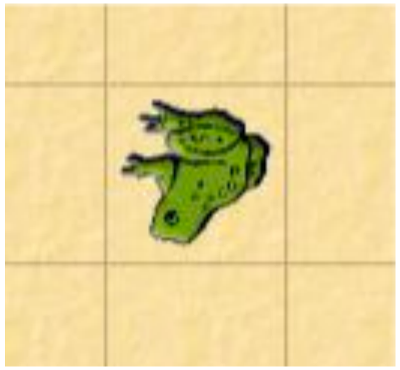
The setRotation() method allows you to tell the object which way to face.



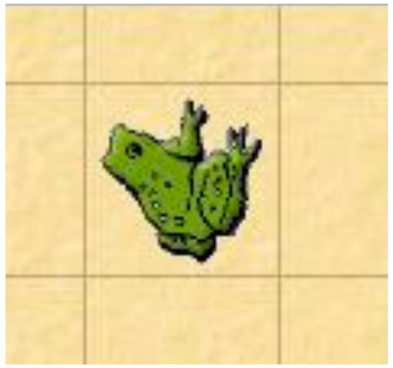
Right = 0o



Down = 90o



Left = 180o



Up = 270o

Using these values inside the setRotation() method will make the frog face the associated direction.

Activity: Program the Actor to Move Using Arrow Keys

Useful code:

Greenfoot.isKeyDown()	”up” , “down” etc.
setRotation()	0, 90, 180, etc.
move()	1 = 1 square, 2 = 2 squares

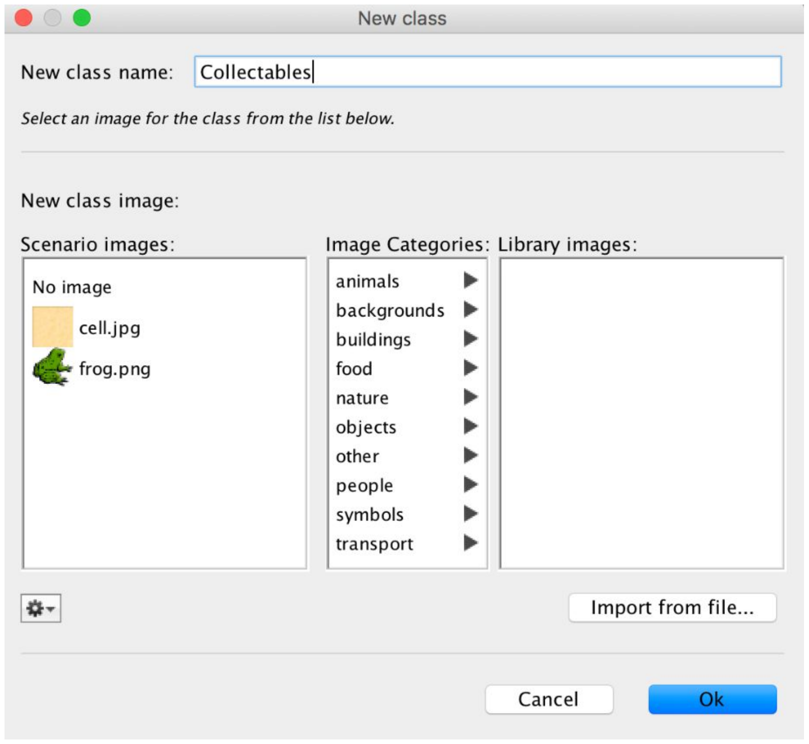
Hint: You can use multiple if statements inside the act() method of the MainCharacter, one below the other.

technocamps

Activity: Adding Collectables

Just as MainCharacter is a subclass of Actor, we want to add another subclass to be a collectable in our game.

Again we need to:
1. Create a new subclass of Actor and name it 'Collectables'
2. Choose the fly image from the animals category


New class

New class name: Collectables

Select an image for the class from the list below.

New class image:

Scenario images: No image, cell.jpg, frog.png

Adding Collectables to the World

Can you remember how to add an object of type Actor to the World?

Add a Collectable to your World.

Hint: First you have to let the World know that you've instantiated a new Actor class called Collectables. Try adapting this:

```
MainCharacter frog = new MainCharacter();
```

You will also need to think of a name for this new object.

Activity: Add 3 Collectables in Various Positions

Using different names, fly1, fly2, fly3, create three more ‘Collectables’ objects.

Note: You do not need another ‘Actor’ subclass. You only need to code three more ‘Collectables’ objects in the ‘World’ programming window.

technocamps

Activity: Frog Eating Collectables

Now we need to write code for when our ‘MainCharacter’ eats our ‘Collectables’.

We need to use the methods:

isTouching(Collectables.class)
removeTouching(Collectables.class)

What do you think these do? Where should we place them?

Once completed, play your program to test that it works.

Sound File available:
http://bit.ly/GreenfootPop

Activity: Adding a Sound

To make this a little more exciting, we will add a sound.

The method:

```
Greenfoot.playSound("filename goes here")
```

will play whichever sound file is named as a string.

You should have a file named pop.wav available to be used which will need to be saved in the **sounds** folder for your project.

Where should you put this method in your code?

Activity: Randomly Moving Collectables

There are a few ways to do this but all will require the use of random numbers.

The method:

```
Greenfoot.getRandomNumber(4)
```

will return a random number between 0 and 3.

Try to combine a random number with the setRotation() method to get a collectable to turn to face up, down, left or right at random

Activity: Adding a Counter

To add a Score counter:

1. Click ‘Edit’ at the top of the screen/window and select the ‘Import Class’ option.
2. Select Counter and press ‘Import’.

Activity: Adding a Counter

The final steps are:

1. Tell the world we're adding a counter named score.
2. Placing it with addObject()
3. Tell our MainCharacter when to add one to the score.

You should remember how to add an object to the world by now!

In our MainCharacter we need to add the following code:

```
Counter score = (Counter) getWorld().getObjects(Counter.class).get(0);
score.add(1);
```

technocamps

Activity: Adding a Counter

```
Counter score = (Counter)getWorld().getObjects(Counter.class).get(0);
score.add(1);
```

The first line is telling the Character which counter we are referring to.

`Counter score = (Counter)`	There is a score of class Counter
`getWorld()`	Returns the world in which the MainCharacter exists
`getObjects(Counter.class)`	Returns a list of all the Counter objects in the world.
`get(0)`	Returns the first item in that list (as there is only one counter)

The second line then simply adds the amount (1) to our score counter using the .add() method which is included in the Counter class's code.

Activity: Negative Collectable

Add another character who will cause the frog to lose a point if it touches this character.

This character will behave differently to our collectables and main character classes and will require you to create a new subclass of actor.

Activity: Two Player Game

Try to develop your game into a two player game.

The second character should be controlled with the W A S D keys.

Try to implement a second counter for the other character that also increases as it collects the collectables.

technocamps

Activity: Quick Quiz

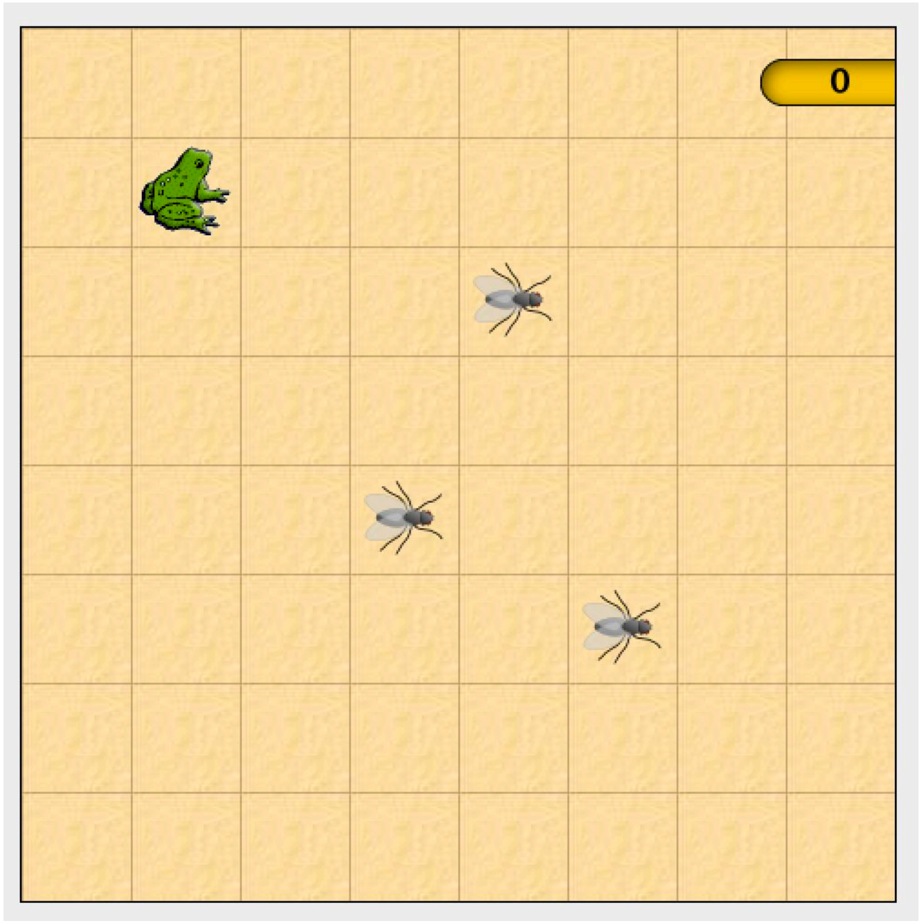
a. How many parent **Classes** are provided by Greenfoot? What are they?

b. How many **Frog Objects** are used in this game?

c. How many **Fly Objects** are used in this game?

d. Is the **Score Counter** an object?

e. How many sub-classes are required for this game?



technocamps

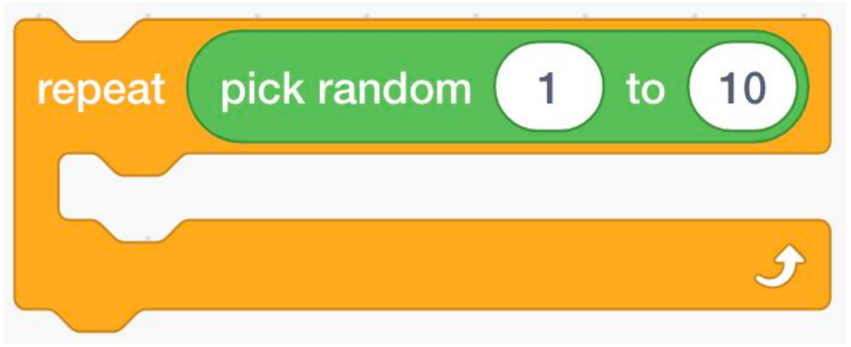
Random Number of Objects

A **loop** in programming is a way to repeat one or more statements.

The **body of the loop** will be repeated while the loop condition is true.

Similar to the repeat block in Scratch, Java has **while** and **for** loops.

We will use **for** loops to create a random number of objects.



technocamps

For Loops

```
for (int i = 0; i < Greenfoot.getRandomNumber(10); i++)
{
        Collectables ant = new Collectables();
        addObject(ant,Greenfoot.getRandomNumber(8),
        Greenfoot.getRandomNumber(8));
}
```

int i = 0	Declaration and initialisation	Declares a number variable i and sets the initial value to 0.
i < Greenfoot.getRandomNumber(10)	Condition	Checks the condition, every time the variable is changed before executing the body of the loop.
i++	Change	Add 1 to the variable i after executing the body of the loop.

Activity: Blow Up The Frog

Every time the MainCharacter, the frog object in our case, eats a Collectable, the fly object, we can increase the size of the frog by a small value.

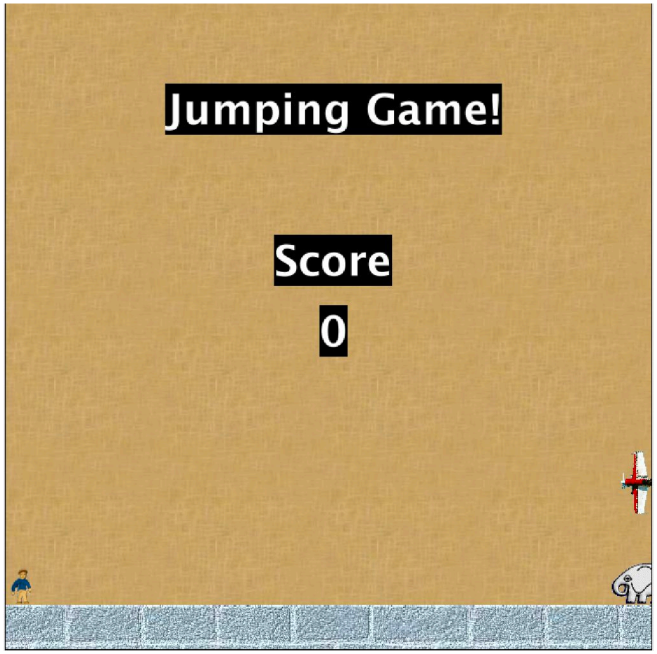
We can do so, by using the Image object associated with each object. All Actor classes have a common method **getImage()** which returns an Image object. The Image class in turn has functions like **getHeight()**, **getWidth()** and **scale(…)** which allows you to get and set the width and height of the image associated with the object.

Check what happens to the frog if you add this line after you have eaten the fly.

```
getImage().scale(getImage().getWidth()+10,
                 getImage().getHeight()+10);
```

Extension: Creating a Jumping Game

You will now be shown a simple game made in Greenfoot which uses jumping and avoiding enemies as the main game mechanic. Your task is to try to create your own game which works In the same way and then develop it yourself.



Creating the World and a Character

Activity:

1. Create World and Character subclasses.
2. Resize the background using the super() method. **64 by 64 cells each with 10 by 10 pixels.**
3. Create a Subclass for Blocks which will act as the floor of the game
4. Place the Character on top of the blocks.
5. Right-Click the screen and Save the World.

After doing this, the starting co-ordinate for our character can be found in the Background code. The y-coordinate is going to be very important!

Setting groundHeight and Adding Movement

First, we need to set our groundHeight for the MainCharacter. This will be the starting y-coordinate.

1. Define our private int groundHeight above the act() method and set it to whatever the y-coordinate is where we placed it.

Remember, in Greenfoot the y-coordinate increases as we move DOWN the screen.

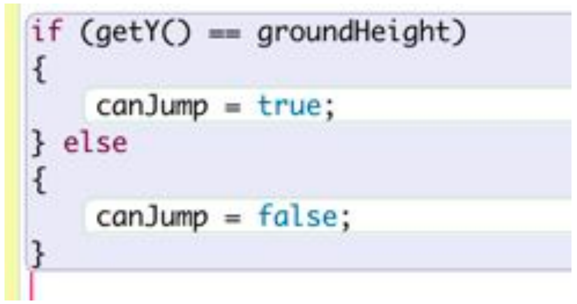
2. Add movement left and right using the move() method and left and right arrow keys.

Jumping

If the up key (or space key) is pressed we want our character to jump.

But we also don't want our character to jump if they are already jumping!

So we need to define a couple of boolean variables and use these:



```
if (getY() == groundHeight)
{
    canJump = true;
} else
{
    canJump = false;
}
```

private boolean canJump = true;

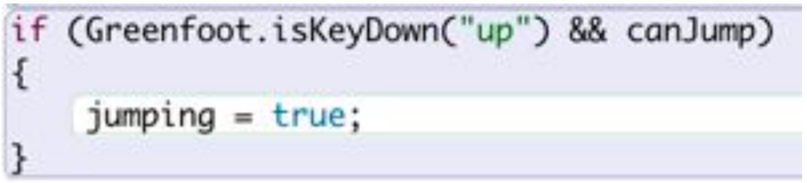
private boolean jumping = false;

i.e. it can only jump, if it is

touching the ground.

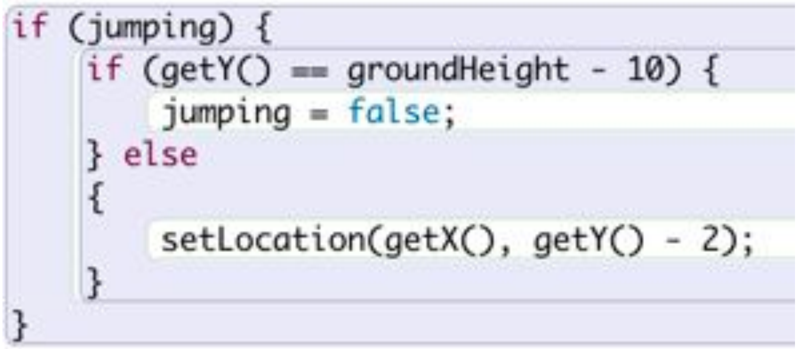
Jumping

So if the character can jump and the up key is pressed then it will be jumping so we set **jumping = true;**



```
if (Greenfoot.isKeyDown("up") && canJump)
{
    jumping = true;
}
```

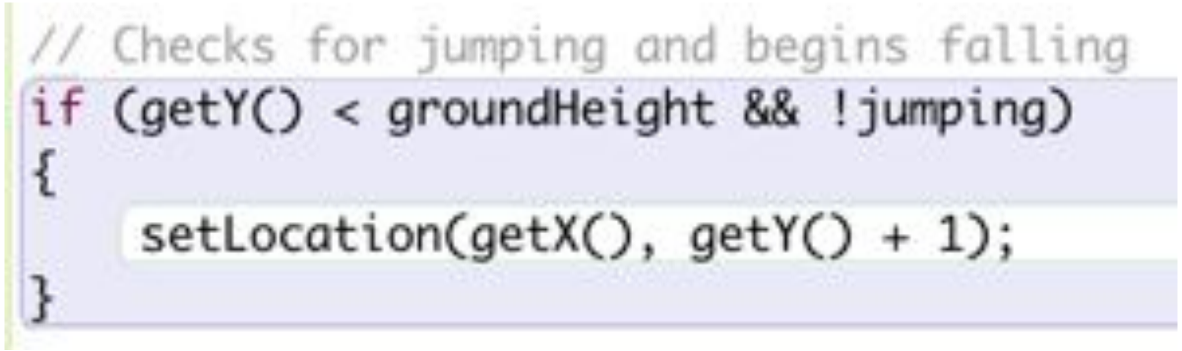
If jumping is true, then we want the character to move upwards to a certain height before stopping.



```
if (jumping) {
    if (getY() == groundHeight - 10) {
        jumping = false;
    } else
    {
        setLocation(getX(), getY() - 2);
    }
}
```

Jumping

Once the character has reached the full height of their jump, we need them to fall back down again. So as they are no longer jumping, but they are not on the ground we can use an if statement to capture this:



```
// Checks for jumping and begins falling
if (getY() < groundHeight && !jumping)
{
    setLocation(getX(), getY() + 1);
}
```

Our character should now be able to jump!

technocamps

Activity: Finish The Game

You should have everything you need to finish the game yourselves now. Think about how the game works and how you would implement that in your game.

If you finish making the game, how could you develop it further? What about adding more obstacles, or maybe adding a power-up which can temporarily stop the enemies?