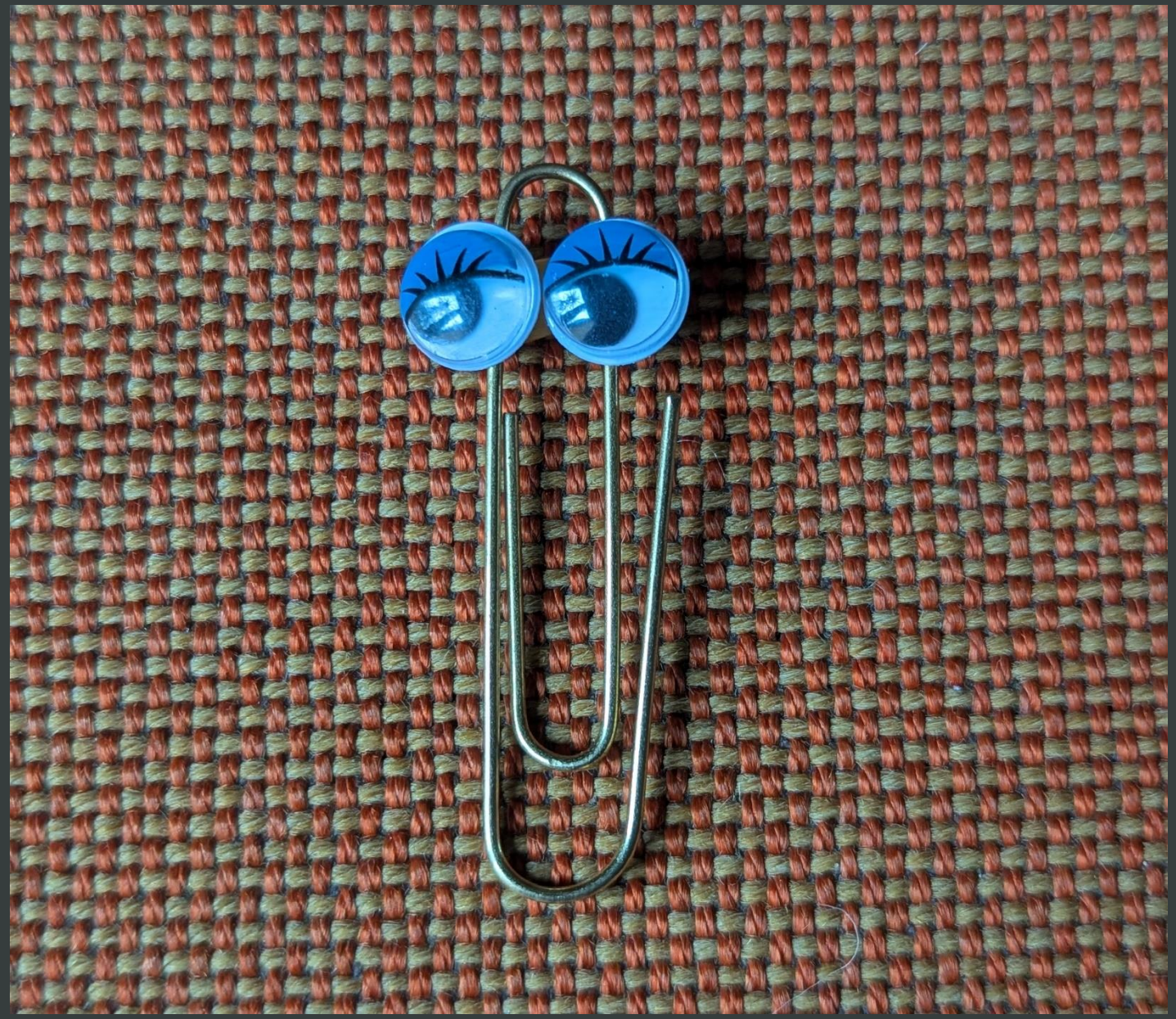
CS21120 Assignment 2024-25

JIM FINNIS
jcf12@aber.ac.uk



Introduction

Oh no! Lots of pages!

- The assignment brief is quite long
- This is to make sure you know exactly what to do
  - But not how to do it!
- Work through it slowly and step by step
  - But look at the whole thing first
- Your code must implement strict Java interfaces – that makes the brief longer
- And there is an introduction to the problem (I'll cover that today too)

Finding rhymes

The problem

List all the words that rhyme with a given word in English, for example

Input:

- security

Output:

- impurity, maturity, obscurity, purity, security, surety

What is a rhyme?

“I must go down to the seas again, to the lonely sea and the sky,

And all I ask is a tall ship and a star to steer her by;

And the wheel’s kick and the wind’s song and the white sail’s shaking,

And a grey mist on the sea’s face, and a grey dawn breaking.”

- The ending of each line sounds the same

(the poem is Sea-Fever, by John Masefield)

What is a rhyme?

“And the wheel’s kick and the wind’s song and the white sail’s sh**a**king,

And a grey mist on the sea’s face, and a grey dawn br**ea**king.”

- **More precisely,** the sounds in both lines are the same **from the last stressed vowel sound onwards**
- (these are the rules I’m working with – there are variations on this, and even more variations for other languages)

Stress?

- The relative emphasis or prominence given to a particular syllable
- In some languages it's predictable (e.g. Polish and Welsh, where it generally goes on the second-to-last syllable)
- In English it's unpredictable and can change the meaning of a word!
  - PRES-ent - a gift
  - pre-SENT - to show, introduce, or give formally
  - CON-tract - a formal agreement
  - con-TRACT - to shrink
- Some words have primary and secondary stress!
  - SHOP-keep-(er) SHOP is primary, keep is secondary, (er) is unstressed

Vowel sound?

- Not to be confused with the vowel letters used to represent those sounds (A,E,I,O,U)
- English has around 20 vowel sounds depending on the variety
- Examples
  - sh**a**king, br**ea**king – both the same sound
  - sea, people, police
  - son, courage, blood
  - foot, put
- We can’t just use English spelling! It’s a mess!

To find rhymes..

- We need to find the **last stressed vowel** in our word
- And compare it with the sounds after the last stressed vowel in all other words
- But to find and compare these sounds we need to know what they are – and we can’t tell from English spelling!

Representing language sounds

- Language sounds are called **phonemes**
- We need to have a way of writing them so we can work with them
- The “standard system” is IPA – but it’s /ɪŋˈkɹɛdɪbli ˈkɒmplɪkeɪtɪd/.
- Instead we’ll use **ARPABET**

ARPABET

- Developed in the 1970s
- For General American English
- Each **phoneme** is either
  - A letter or pair of letters for a **consonant sound**
  - A letter or pair of letters and a number (0, 1, 2) for a **vowel sound**
- Examples (I've underlined vowels and put them in orange, and put consonants in blue):
  - cat K AE1 T
  - nightshade N AY1 CH EY2 D
  - anything EH1 N IY0 TH IH2 NG

ARPABET chart

AA	f(a)ther	**UH**	b(oo)k	**N**	(n)ap
AE	b(a)t	**UW**	b(oo)t	**NG**	so(ng)
AH	b(u)t	**B**	(b)at	**P**	(p)at
AO	c(au)ght, st(o)ry	**CH**	(ch)in	**R**	(r)ed
AW	(ou)t	**D**	(d)og	**S**	(s)ee
AY	b(i)te	**DH**	(th)is	**SH**	(sh)oe
EH	b(e)t	**F**	(f)ish	**T**	(t)ap
ER	b(i)rd	**G**	(g)o	**TH**	(th)ink
EY	b(ai)t	**HH**	(h)at	**V**	(v)an
IH	b(i)t	**JH**	(j)am	**W**	(w)et
IY	b(ee)t	**K**	(c)at	**Y**	(y)es
OW	b(oa)t, c(o)ne	**L**	(l)ight	**Z**	(z)oo
OY	b(oy)	**M**	(m)ap	**ZH**	mea(s)ure

Vowels (in blue) must be followed by a stress number

- 1 for primary stress (loudest)
- 2 for secondary stress
- 0 for unstressed

shopkeeper: SH AA1 P K IY2 P ER0

The CMU Dictionary

- How can we get the ARPABET pronunciation for a word?
- The **Carnegie-Mellon University Pronouncing Dictionary** (or “cmudict”)
  - Has 135000+ pronunciations for 126000+ different words
  - Some words have more than one pronunciation
  - Is a plain text file
  - Some of the words are very obscure (and there are a lot of names)
  - There are some mistakes (“prokofiev P R AA1 K OW0 F IY2 V” – “PROK-oh-feev” No.)

The CMU Dictionary – an excerpt

```
tao T AW1
tao(2) D AW1
taoiseach T IY1 SH AH0 K # title, irish
taoiseach's T IY1 SH AH0 K S
taoism D AW1 IH0 Z AH0 M
taoist D AW1 IH0 S T
taoists D AW1 AH0 S T S
taormina T AA0 AO0 R M IY1 N AH0
```

The CMU Dictionary format

```
dail(2) D OY1 L # org, irish
```

`dail`	Word
`(2)`	Optional pronunciation number in brackets (no space!)
	Space
`D OY1 L`	Phonemes separated by spaces
`# org, irish`	Optional #-sign followed by comment

Then the problem is

- Given a word, find it in the CMU dictionary
- For every pronunciation compare it against every other pronunciation of every word
  - Find the last stressed vowel in both pronunciations
  - If the last stressed vowel and all following phonemes are the same in both (ignoring stress), the words rhyme – add it to the output

The tasks

Interfaces

- I will provide **interfaces**
- Your code **must** implement the interfaces I provide
- You **must not modify the interfaces**

DO NOT CHANGE MY INTERFACES

- Someone does it every year.
- **It stops my tests working.**
- **Make sure you get the constructors right too!**

Interfaces

```
public interface IStackOfIntegers {
    void push(int i);
    int pop();

    boolean isEmpty();
    boolean isFull();
}
```

```
public class MyIntegerStack implements IStackOfIntegers {
    private int[] stack;
    private int top;

    public MyIntegerStack(int size) {
        stack = new int[size];
        top = -1;
    }

    public void push(int i) {
        stack[++top] = i;
    }

    public int pop() {
        if (isEmpty()) {
            throw new IllegalStateException("Stack is empty");
        }
        return stack[top--];
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public boolean isFull() {
        return top == stack.length - 1;
    }
}
```

Interfaces

```
public interface IStackOfIntegers {
    void push(int i);
    int pop();

    boolean isEmpty();
    boolean isFull();
}
```

What a class should do

One possible way of doing it

(includes free bug)

```
public class MyIntegerStack implements IStackOfIntegers {
    private int[] stack;
    private int top;

    public MyIntegerStack(int size) {
        stack = new int[size];
        top = -1;
    }

    public void push(int i) {
        stack[++top] = i;
    }

    public int pop() {
        if (isEmpty()) {
            throw new IllegalStateException("Stack is empty");
        }
        return stack[top--];
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public boolean isFull() {
        return top == stack.length - 1;
    }
}
```

Interfaces and Classes

Interfaces

- Contain method signatures, not actual code that does stuff
- Describe how a class should behave
- So multiple classes can “slot” into code expecting that interface
- An interface is a **contract** saying that a class promises to implement certain methods

Classes

- Implement interfaces
- Must then implement the methods in their interfaces
- Contain actual code that does stuff

Tasks and Marks

- Task 1: The Phoneme class (10%)
  - Represents phonemes in a pronunciation –ARPABET values and stress (if a vowel)
- Task 2: The Pronunciation class (20%)
  - Represents a sequence of phonemes, describing how a word is pronounced
- Task 3: The Word class (10%)
  - Represents a word, with its possible pronunciations
- Task 4: The Dictionary class (25%)
  - Represents a collection of words – has code for reading from a file
- Task 5: Rhyming (25%)
  - Completing the Pronunciation and Dictionary classes to detect rhymes

Preparing

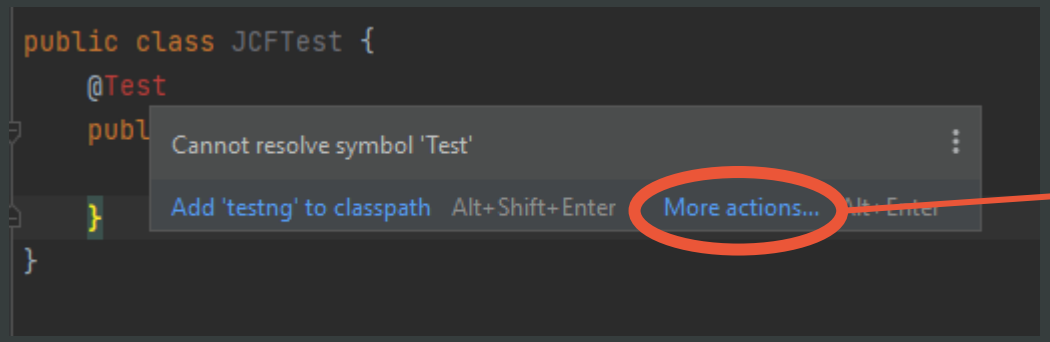
- Create a new project, and inside it create three packages:
  - uk.ac.aber.cs21120.rhymes.interfaces
  - uk.ac.aber.cs21120.rhymes.solution
  - uk.ac.aber.cs21120.rhymes.tests
- Code provided in CS21120_ProvidedCode.zip
- Copy all the files in the “interfaces” directory inside that Zip archive into your uk.ac.aber.cs21120.rhymes.interfaces package
- These are the interfaces which describe how your code will work.
- **Do not copy the tests into your new tests package until you need them!**

Testing your work

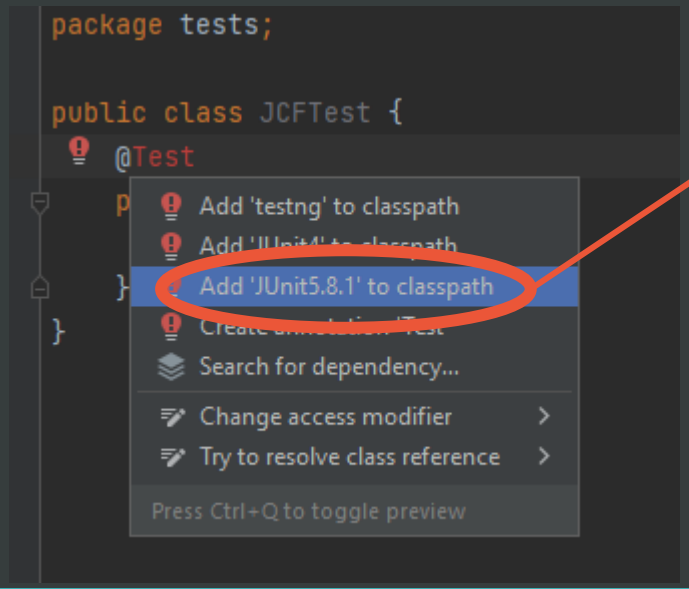
- **There is no main class**
  - Instead, use JUnit tests to run and debug the code.
  - There is a separate set of tests for each class
  - Your code **must implement provided interfaces** for the tests to work!
  - **I will use the tests myself when marking!**

JUnit 5, not JUnit 4!

Error on @Test – hover over it, you will see this popup:



- Click on More actions
- select JUnit 5



You will then see a green arrow icon next to each test, and at the top of each test class. Click this to run a test or entire test class.

Provided code: Arpabet

- In addition to interfaces, the interface package contains the **Arpabet** enum
  - (quick reminder on enums: https://jenkov.com/tutorials/java/enums.html)
- Has a value for each phoneme (but NOT including stress)
- For example, **Arpabet.AH, Arpabet.K**
- To turn a string into an Arpabet value, use **Arpabet.valueOf(String s)**
- To test if the value is a vowel use the boolean **isVowel()** method

Task 1: implementing IPhoneme

- A phoneme inside a word
- Consists of an Arpabet enum value and a stress (or -1 if the Arpabet value is not a vowel)
- This **must** be in the **uk.ac.aber.cs21120.rhymes.solution** package
- It **must** be called **Phoneme**
- It **must** implement the methods in the **IPhoneme** interface in uk.ac.aber.cs21120.rhymes.interfaces:
  - **Arpabet getArpabet()**
  - **int getStress()**
  - **boolean hasSameArpabet(IPhoneme other)**
- It **must also** implement a constructor of the form **Phoneme(Arpabet phoneme, int stress).**

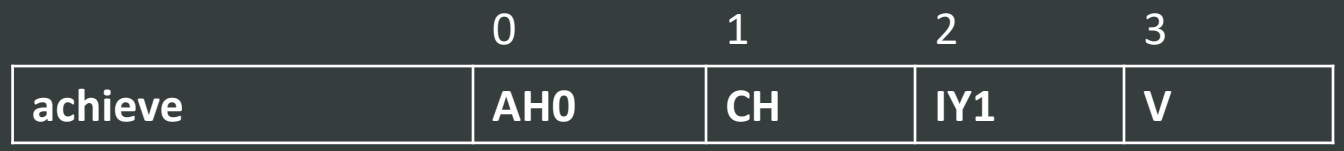
Task 2: implementing IPronunciation

- A sequence of phonemes (IPhoneme objects)
- Again, must be in the right package with the right name and implementing the right methods
- Easy methods:
  - **add(IPhoneme p)**
  - **List<IPhoneme> getPhonemes()**
  - **boolean rhymesWith(IPronunciation other)** – Leave as a stub for now (until task 5)
- Hard method:
  - **int findFinalStressedVowelIndex() …**

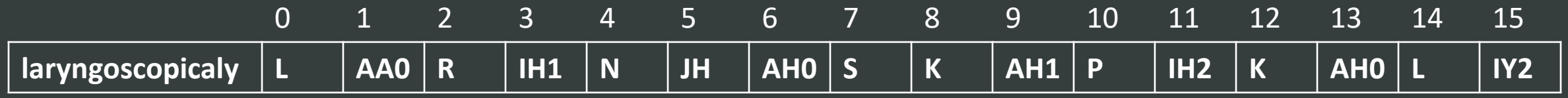
Task 2: findFinalStressedVowelIndex

Find the **location** of the last vowel with the highest stress:

- The last vowel with primary stress if there is one (stress=1)
- If there is no vowel with primary stress, the last vowel with secondary stress (stress=2)
- If there is no vowel with secondary stress, the last vowel of any kind
- -1 if there are no vowels at all.


0 1 2 3
achieve AH0 CH IY1 V


0 1 2 3 4 5 6 7 8 9 10
asymptomatic EY2 S IM2 P T AH0 M AE1 T IH0 K


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
laryngoscopicaly L AA0 R IH1 N JH AH0 S K AH1 P IH2 K AH0 L IY2

	0	1	2	3
achieve	AH0	CH	IY1	V

Word	LSV
achieve	2
asymptomatic	7
laryngoscopicaly	9

	0	1	2	3	4	5	6	7	8	9	10
asymptomatic	EY2	S	IM2	P	T	AH0	M	AE1	T	IH0	K

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
laryngoscopicaly	L	AA0	R	IH1	N	JH	AH0	S	K	AH1	P	IH2	K	AH0	L	IY2

Task 3: Implementing IWord

- A string (the English spelling) and a collection of pronunciations
- Must be in the right package with the right name (Word)
- Must have the methods
  - **String getWord()**
  - **addPronunciation(IPronunciation p)**
  - **Set<IPronunciation> getPronunciations()**
- And the constructor
  - **Word(String word)**

Task 4: Implementing IDictionary and reading data

- Write **Dictionary** in the correct package. It must implement **IDictionary**.
- Stage 1 – write and test
  - **IWord getWord(String s)**
  - **void addWord(IWord w)**
  - **int getWordCount()**
  - **int getPronunciationCount()**
  - but write **parseDictionaryLine** and **loadDictionary** as “stubs” only (empty methods)

Task 4: Implementing IDictionary and reading data

- Next write **parseDictionaryLine(String line)**
- This takes a line of text and processes it, adding pronunciation and perhaps a new word to the dictionary

```
dail(2) D OY1 L # org, irish
```

- Now write **loadDictionary(String filename)**
- This opens a file and processes all the lines.
- You can get the file from https://users.aber.ac.uk/jcf12/downloads/cmudict.dict
  - use right click and Save As…

Task 5: Rhyming

- Stage 1 – complete **Pronunciation.rhymesWith**
  - This compares the pronunciation with another
  - They are the same if their phonemes contain the same Arpabet values from their last stressed vowels onwards.

	0	1	2	3
achieve	**AH0**	**CH**	**IY1**	**V**

	0	1	2
heave	**CH**	**IY1**	**V**

Task 5: Rhyming

- Stage 2 – complete **Dictionary.getRhymes**
  - This compares all pronunciations of a given words with every pronunciation of every word
  - Returns a set of the English spellings of all words which rhyme

Common mistakes

- Changing an interface
- Submitting .class files instead of .java files
- Putting code into the wrong packages (should be **uk.ac.aber.cs21120.rhymes.solution**)
- Using the wrong constructor
- Making something **static** without need (because you don't understand **static**)
- Not running my unit tests
- Not using a debugger (https://www.youtube.com/watch?v=lAWnIP1S6UA)
- Submitting a .docx (Word document) and not a PDF (I'm on Linux!)
- Submitting a RAR or some other weird archive and not a ZIP file

How I’ll be marking

- I’ll read the code and your report, obviously!
- **But first I will run the code through an automated test suite.**
- Make sure you have used all the correct interfaces and put things into packages with the right names **or my tests will not work.**

Final words

- **Revise your Java**. Particularly the difference between classes, objects and references (this might help: https://users.aber.ac.uk/jcf12/teaching/cs123/examples/)
- **Do as much as you can**. The early parts of the assignment carry a lot of marks.
- **Don’t worry that the assignment looks very long!** I’ve added a lot of detail to guide you through the work. Just take it step by step.
- **Please, please, please don’t copy each other’s code or code from the Internet.** We have software which detects this, and it’s very effective.

Questions?

Email: **jcf12@aber.ac.uk**

Office hours

- 12:00 until 13:00 on Thursdays
- 10:00 until 11:00 on Fridays

These may change, but up to date hours are on https://users.aber.ac.uk/jcf12/