Past-paper walk-through

Abstraction and Program Design ▪ 3.1

eXercise

Questions in this set

- 2015 Q4

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 4

2015 ▪ Q4

This question involves the design of an interface, writing a class that implements the interface, and writing a method that uses the interface.

(a) A *number group* represents a group of integers defined in some way. It could be empty, or it could contain one or more integers.

Write an interface named `NumberGroup` that represents a group of integers. The interface should have a single `contains` method that determines if a given integer is in the group. For example, if `group1` is of type `NumberGroup`, and it contains only the two numbers -5 and 3, then `group1.contains(-5)` would return `true`, and `group1.contains(2)` would return `false`.

Write the complete `NumberGroup` interface. It must have exactly one method.

Question 4

Call	Result
`multiple1.contains(2)`	`true`
`multiple1.contains(9)`	`false`
`multiple1.contains(6)`	`true`

Question 4

2015 ▪ Q4

This question involves the design of an interface, writing a class that implements the interface, and writing a method that uses the interface.

(b) A *range* represents a number group that contains all (and only) the integers between a minimum value and a maximum value, inclusive.

Write the `Range` class, which is a `NumberGroup`. The `Range` class represents the group of `int` values that range from a given minimum value up through a given maximum value, inclusive. For example, the declaration

```
nNumberGroup+w nrange1+w o=+w knew+w nRangep(ol+m+mi3p,+w l+m+mi2p)p;
```

Question 4

represents the group of integer values -3, -2, -1, 0, 1, 2.
Write the complete `Range` class. Include all necessary instance variables and methods as well as a constructor that takes two `int` parameters. The first parameter represents the minimum value, and the second parameter represents the maximum value of the range. You may assume that the minimum is less than or equal to the maximum.

Question 4

2015 ▪ Q4

This question involves the design of an interface, writing a class that implements the interface, and writing a method that uses the interface.

(c) The `MultipleGroups` class (not shown) represents a collection of `NumberGroup` objects and is a `NumberGroup`. The `MultipleGroups` class stores the number groups in the instance variable `groupList` (shown below), which is initialized in the constructor.

```
k+kdprivate+w nListonNumberGroupo+w ngroupListp;
```

Question 4

Write the `MultipleGroups` method `contains`. The method takes an integer and returns `true` if and only if the integer is contained in one or more of the number groups in `groupList`. For example, suppose `multiple1` has been declared as an instance of `MultipleGroups` and consists of the three ranges created by the calls `new Range(5, 8)`, `new Range(10, 12)`, and `new Range(1, 6)`. The following table shows the results of several calls to `contains`.

Call	**Result**
`multiple1.contains(2)`	`true`
`multiple1.contains(9)`	`false`
`multiple1.contains(6)`	`true`

Question 4

Complete method contains below.

```
c+cm/** Returns true if at least one of the number groups in this multiple
c+cm *          false otherwise.
c+cm */
k+kdpublic+w k+ktboolean+w n+nfcontainsp(k+ktint+w nnump)
```