W11 Practical By 180003815

Access Extension via

```
git checkout Extension1
```

Overview

Initial Specification

The specification requires that a software system for a company which sells holidays is implemented.

Description of Scenario

There will be three types of holiday; adventure, relaxation and culture. Each holiday will have the attributes:

- unique id
- location
- cost (will be stored in pence)
- starting date
- duration (assuming it is days)

Adventure holidays have a difficulty level. Culture holidays have aspects of interest and may also contain a lecture on these aspects. Relaxation holidays have a description of amenities available (It is assumed this is a single text description and not a list of text description).

The company has employees which have attributes:

- staff id
- name
- salary (will be stored in pence)

There are two categories of staff members holiday guides and admin staff. Holiday guides specialise in an activity(1 activity). Admin staff have one or more skills.

Each holiday has a member of admin staff associated with it. In addition, each adventure holiday has a holiday guide associated with it.

Required Functionality

- Print the names and salaries of all staff of the company in a tidy format of your choice
- Calculate the total salary cost of the company;
- Give all staff a salary increment of a given percentage;
- Print the names and specialisations of all holiday guides (assume it does not mean exclusively print them alone);
- Print the id, location and difficulty level of all of adventure holidays that are offered by the company and the names of the guides associated with them;
- Print the details of all holidays offered by the company that fall within a date range;

- Find the most expensive holiday offered by the company;
- Print the details of all culture holidays which include a lecture

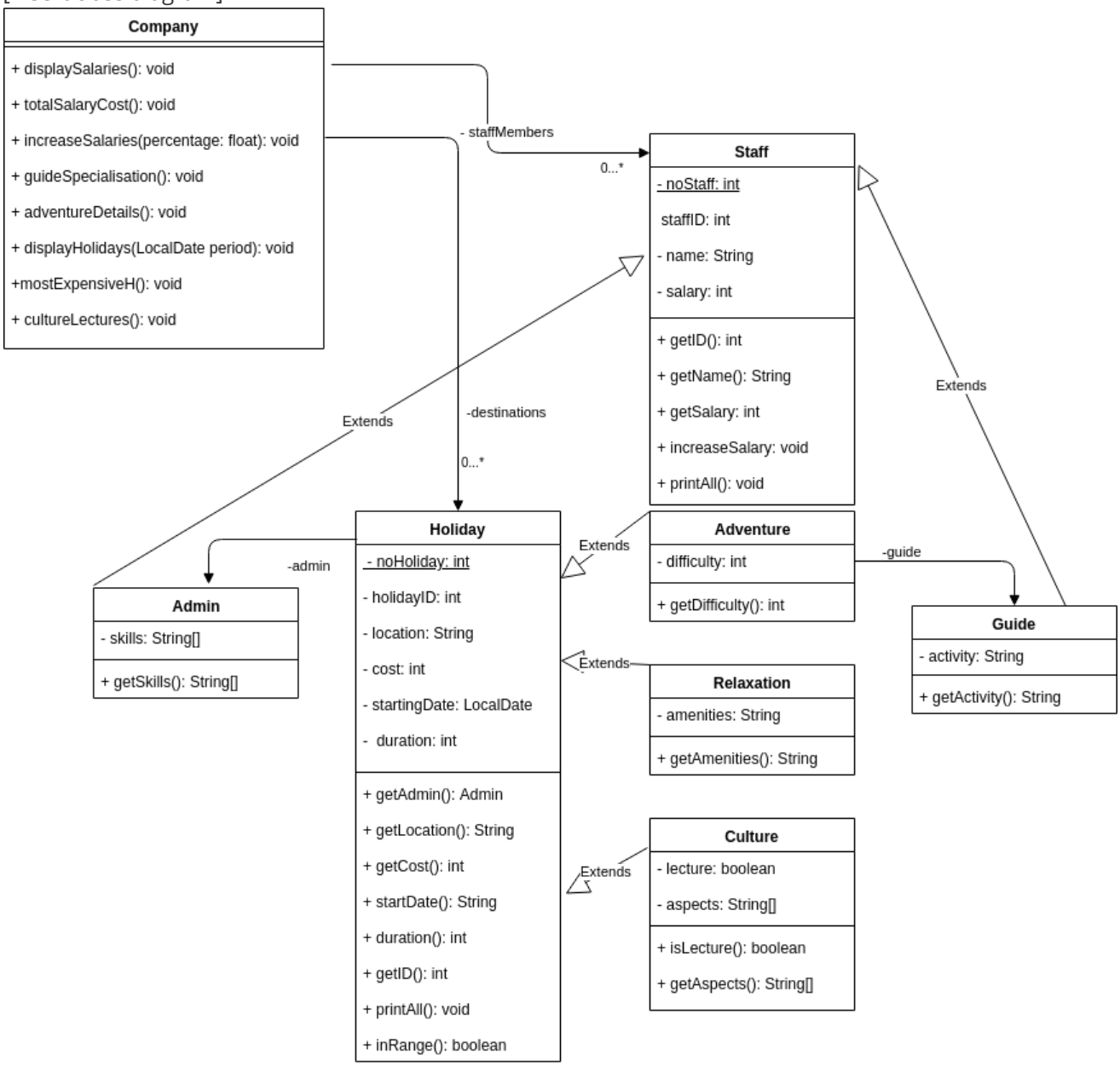
Extension 1: ENUMS

These allows you to have pre-defined types in java, because every ENUM variable must point to An Enum constant.

Design

Initial Specification

[insert class diagram]



Justification of Structure

Admin and Guide have features in common both as both are Members of Staff. They are a subtype of staff. So both will have salaries, names and staff ids. So the behaviours they share in common have been inserted into a single class which they both inherit. Then the attributes they shall declared in their respective classes.

Adventure, Culture and Relaxation Holidays are a type of holiday. So generalised model of what holiday contains was made and then the subtypes. So Classes Adventure, Culture and Relaxation shall all inherit from the Holiday Superclass

It was decided since the values of the instance variables of the superclass of both generalisations would not be changed during run time, they could be made private and they could have getter which return their values. The only instance variable that is required to be changed during runtime is the salary variable but by a particular percentage. So a method will be created to allow for this.

The functions which the program is required to have have be made into methods of a class called company which represent the holiday company. These functions represent the different use cases of the program and the different actions the user many perform. These methods will be return void as they are only requested to print values or to update values. The company class will store all the employs and the destinations. The company class will have a one to many associations with Holiday and Staff class since the contain will have a certain number of employees working for them and set of destinations for users to book. This will allow for subtypes of the superclass's to be stored in a single array.

`Company`

The arrays `staffMembers[]` and `destinations[]` which stores the list of staff members and holiday destinations are keep private as it assumed that they are not required to be changed during runtime as it has not been explicitly required in the specification. So since the program is not required to add or remove destinations or employees then to avoid miss management of variable they shall be keep private.

`Staff`

A static variable `noStaff` is used to create unique ID as it is independent of the current instance of a object so can be increments when a new object is created. `increaseSalary` allows for the incremental increase of the value of salary. There exists a `printAll()` method inside of the `Staff` class because it allows for the printing of a the data contained within the object. Having it in this class prevents the contained from being explicitly written was time in its sub class or in the functions of company class that require the printing of all data within the object. This only use `System.out.print()` which does not print a new line so it sub classes can print there own data added onto it.

`Admin`

Inherits all attributes from Staff class. It assume the skills of an admin are variable so they are of type array with no predefined length. Contains `printAll()` method which makes use of overriding . This method calls the `printAll()` method of it superclass to run first then adds on the rest of admin data which was not include in its `printAll()` method and makes use of `System.out.println()` to make sure a new line.

`Guide`

It is assume that there can only be one specialised activity so the type of `activity` is just `String`. `printAll()` makes uses of overriding with the same reasoning as `Admin` class.

`Holiday`

`noHolidays` is static with same reasoning as `noStaff`. the variable `admin` is type `Admin` so there is an actual association between the 2 classes. So the methods of the admin class can be accessed from within the holiday such as in `printAll()` where it is use to call `getName()` method. `printAll()` method with exists for the same logic and is implemented with the same logic as in the `Staff` class. `inRange()`.

`Relaxation`

inherits from `Holiday`. Only different in that it contains a new attribute called `amenties`. Which requires a override `printAll()` method to be displayed with rest of all the data.

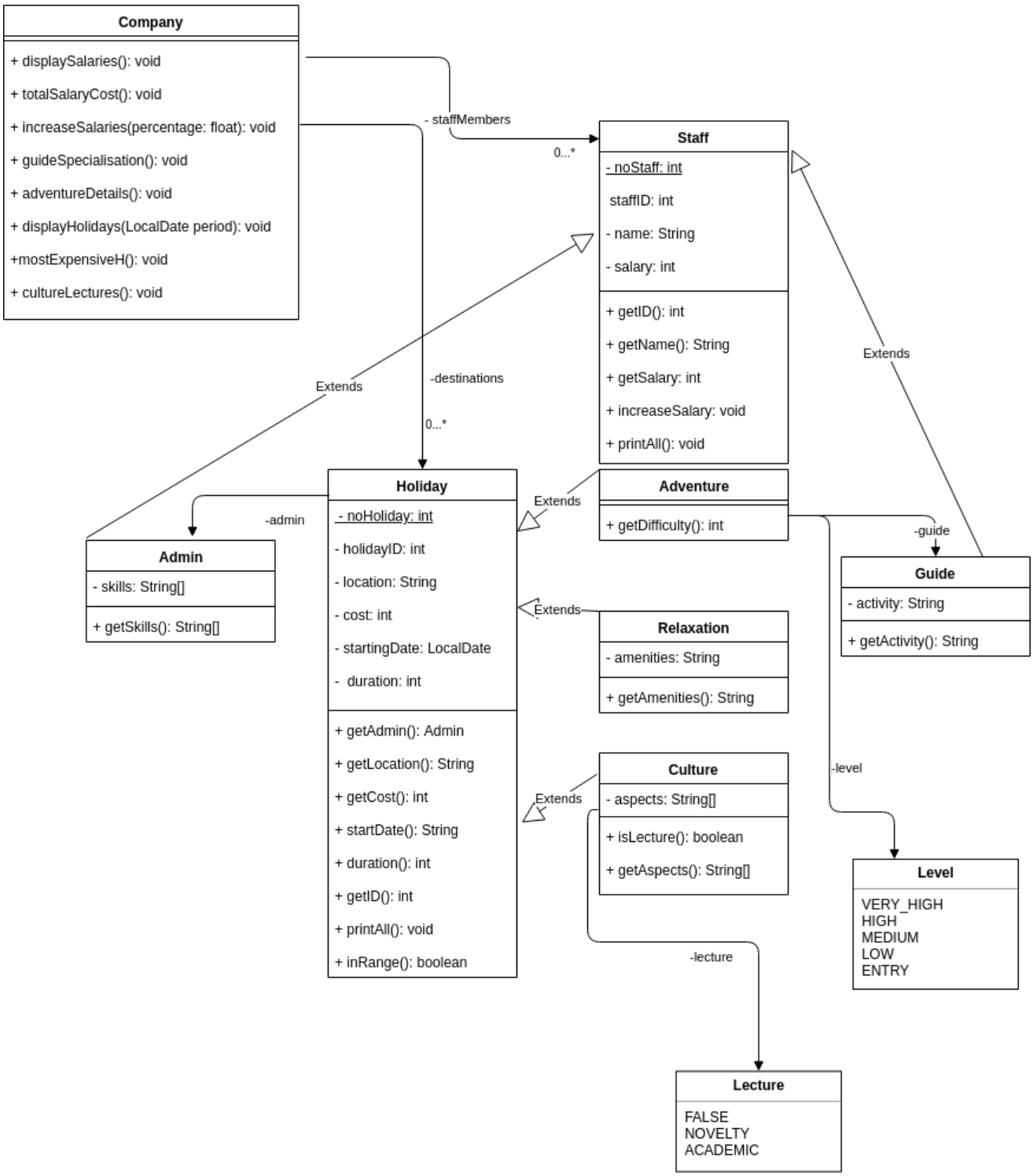
`Culture`

Inherits all the attributes of `Holiday` class. A holiday either has a lecture or it doesn't so the type of `lecture` is `boolean`. It as assume that the number of aspects can be variable so it has been made an array.

`Adventure`

Inherits all the attributes of `Holiday` class. The `guide` is of type `Guide` with same reasoning as admin. difficulty is set on a numbered scale so `difficulty` is of type `int`.

Extension 1



Levels are now described as enums

Lectures are now described as enums

Testing

To test the solution a test scenario shall be devised where 30 holidays shall be created 10 of each type and 3 admins shall exists and 10 guides. All employees shall have the exact same salary 100. So all should be effected in same way by salary increase. Each of the functionality methods shall be ran on the program to determine if it matches the desired output.

Test 1

13 Employees are randomly generated. There Name and Salary are displayed. Then the total Cost of there salary is calculated. Since they all have the same salary of £100 they the total cost should be £1300. Then there will be an percentage increase of 15% so the prices of every salary will increase so the £1300 total is expected to increase by 15% as well.

Expected output

13 employees and there wage £100 displayed.

Total cost of £1300 displayed.

13 employees and there wage £150 displayed.

Total cost of £1950 displayed.

Actual Output

```
---Employee: Salary(£)---
Guide 0: £100.0
Guide 1: £100.0
Guide 2: £100.0
Guide 3: £100.0
Guide 4: £100.0
Guide 5: £100.0
Guide 6: £100.0
Guide 7: £100.0
Guide 8: £100.0
Guide 9: £100.0
Admin 10: £100.0
Admin 11: £100.0
Admin 12: £100.0
---Total Salary Cost: £1300.0
---Employee: Salary(£)---
Guide 0: £150.0
Guide 1: £150.0
Guide 2: £150.0
Guide 3: £150.0
Guide 4: £150.0
Guide 5: £150.0
Guide 6: £150.0
Guide 7: £150.0
Guide 8: £150.0
Guide 9: £150.0
Admin 10: £150.0
Admin 11: £150.0
Admin 12: £150.0
---Total Salary Cost: £1950.0
```

Test 2

30 Holidays are randomly generated with costs and random dates. The first holiday in the random array of holidays is taken and it date passed as the argument of the `displayHolidaysIn()`. Then the most `displayHolidaysIn()`.

Expected output

The First Random generated holiday

The Holidays are over date of randomly generated holiday So from the date of found holiday plus the days of the duration.

printing of only holiday that satisfy the condition.

Actual Output

```
First randomly generated Holiday: 2019-01-13
Holidays that are over 2019-01-13
|0|Admin 10|Adventure Location 0| £737.96|2019-01-13|10|Guide 0|0|
|9|Admin 10|Adventure Location 3| £4142.65|2019-01-07|13|Guide 3|3|
|17|Admin 12|Relaxation Location 5| £3017.16|2019-01-08|21|amenities: 5|
---------MOST EXPENSIVE HOLIDAY!!!---------
|1|Admin 11|Culture Location 0| £4575.23|2019-08-16|11|true|Aspect 0, Aspect 0|
```

Test 3

The following methods are ran

- `adventureDetails`
- `guide Specilisation`
- `cultureLectures`

Expected Output

The details of 10 random adventure objects are printed.

The details of 10 random guide objects are printed.

Since the condition for the lecture to be true when generating culture holidays was `i%2 ==0` then only even culture holiday should appear when their details are printed.

Actual Output

```
|0|Admin 10|Adventure Location 0| £2758.46|2019-05-19|9|Guide 0|0|
|3|Admin 10|Adventure Location 1| £4910.78|2019-01-17|11|Guide 1|1|
|6|Admin 10|Adventure Location 2| £3515.27|2019-09-28|3|Guide 2|2|
|9|Admin 10|Adventure Location 3| £3360.82|2019-03-08|3|Guide 3|3|
|12|Admin 10|Adventure Location 4| £1640.16|2019-05-26|9|Guide 4|4|
|15|Admin 10|Adventure Location 5| £3493.13|2019-02-23|8|Guide 5|5|
|18|Admin 10|Adventure Location 6| £2587.07|2019-01-20|12|Guide 6|6|
|21|Admin 10|Adventure Location 7| £605.65|2019-10-26|12|Guide 7|7|
|24|Admin 10|Adventure Location 8| £3354.11|2019-10-17|7|Guide 8|8|
|27|Admin 10|Adventure Location 9| £1986.91|2019-05-12|16|Guide 9|9|
----------------------------------------
---Holiday Guide: Specilisation---
|0|Guide 0|10000|Activity 0|
|1|Guide 1|10000|Activity 1|
|2|Guide 2|10000|Activity 2|
|3|Guide 3|10000|Activity 3|
|4|Guide 4|10000|Activity 4|
|5|Guide 5|10000|Activity 5|
|6|Guide 6|10000|Activity 6|
|7|Guide 7|10000|Activity 7|
|8|Guide 8|10000|Activity 8|
|9|Guide 9|10000|Activity 9|
----------------------------------------
---Holiday's With Lectures---
|1|Admin 11|Culture Location 0| £1513.76|2019-11-26|3|true|Aspect 0, Aspect 0|
|7|Admin 11|Culture Location 2| £2284.99|2019-11-19|4|true|Aspect 2, Aspect 4|
|13|Admin 11|Culture Location 4| £1543.84|2019-08-27|12|true|Aspect 4, Aspect 8|
|19|Admin 11|Culture Location 6| £2656.68|2019-12-09|9|true|Aspect 6, Aspect 12|
|25|Admin 11|Culture Location 8| £579.19|2019-04-07|5|true|Aspect 8, Aspect 16|
```

Extension 1

The Test 3 is done again.

Expected Output

enums replace values

Oscilation between different Level values

Oscillation between different Lecture values excluding FALSE

Actual Output

```
|0|Admin 10|Adventure Location 0| £3657.1|2019-09-09|4|Guide 0|ENTRY|
|3|Admin 10|Adventure Location 1| £4543.8|2019-12-10|16|Guide 1|LOW|
|6|Admin 10|Adventure Location 2| £3649.54|2019-01-25|3|Guide 2|MEDIUM|
|9|Admin 10|Adventure Location 3| £1483.3|2019-04-14|19|Guide 3|HIGH|
|12|Admin 10|Adventure Location 4| £1524.78|2019-08-18|21|Guide 4|VERY_HIGH|
|15|Admin 10|Adventure Location 5| £380.82|2019-12-06|20|Guide 5|ENTRY|
|18|Admin 10|Adventure Location 6| £2782.88|2019-10-02|19|Guide 6|LOW|
|21|Admin 10|Adventure Location 7| £4855.75|2019-04-02|18|Guide 7|MEDIUM|
|24|Admin 10|Adventure Location 8| £4116.63|2019-05-12|19|Guide 8|HIGH|
|27|Admin 10|Adventure Location 9| £940.69|2019-10-17|19|Guide 9|VERY_HIGH|
----------------------------------------
---Holiday Guide: Specilisation---
|0|Guide 0|10000|Activity 0|
|1|Guide 1|10000|Activity 1|
|2|Guide 2|10000|Activity 2|
|3|Guide 3|10000|Activity 3|
|4|Guide 4|10000|Activity 4|
|5|Guide 5|10000|Activity 5|
|6|Guide 6|10000|Activity 6|
|7|Guide 7|10000|Activity 7|
|8|Guide 8|10000|Activity 8|
|9|Guide 9|10000|Activity 9|
----------------------------------------
---Holiday's With Lectures---
|1|Admin 11|Culture Location 0| £136.13|2019-11-24|18|ACADEMIC|Aspect 0, Aspect 0|
|7|Admin 11|Culture Location 2| £4735.42|2019-02-26|8|NOVELTY|Aspect 2, Aspect 4|
|13|Admin 11|Culture Location 4| £1231.1|2019-02-05|17|ACADEMIC|Aspect 4, Aspect 8|
|19|Admin 11|Culture Location 6| £2454.9|2019-10-26|18|NOVELTY|Aspect 6, Aspect 12|
|25|Admin 11|Culture Location 8| £4858.18|2019-09-01|16|ACADEMIC|Aspect 8, Aspect 16|
```

Research

Activity Diagram

An activity Diagram is a UML behaviour Diagram, it is a graphical representation of stepwise refinement activities, show how different activities can be coordinated at a certain level of abstraction:

- Demonstrate the logical structure of a software solution
- Describe the different steps in a UML use case.

The different components of an Activity Diagram:

- **Action:** Where the the users or software perform a step in activity.
- **Decision:** Represent by a diamond; conditional branch.
- **Control Flows:** connectors that denotes the flow between steps.
- **Start Node:** Denotes the beginning of an activity
- **End Node:** Denotes the final step in an activity

Source: https://www.lucidchart.com/pages/uml-activity-diagram

Use Case Diagram

This is a UML Design methodology used to display the possible different ways user of a system can interact with that system. A use case diagrams consist of:

- **Use Cases:** A specific actions that can be undertook by user when they interact with a system.
- **Actors:** A specific role which is played the user that interacts with a system.

- **Boundary:** defines the system and separates it from the world around it.
- **Relationships:** Lines which indicate the relationship of a Actor to a use case.

source: https://www.visual-paradigm.com/guide/uml-unified-modelling-language/what-is-actvity-diagram/

Evaluation

Initial Specification

It was shown through testing that the specification was met as all the methods which execute the functionality required by the program indeed showed the correct output when there results were obtained.

As correct outputs were given from testing

- Test 1: proved the functionality of `displaySalaries()`, `totalSalaries()` and `increaseSalaries()`
- Test 2: proved the functionality of `displayHolidaysIn()` and `mostExpensiveH()`
- Test 3: proved the functionality of `guideSpecialisation()` and `cultureLectures()`

Since these all correlated to required functions of the specifications the programs requires have been fully satisfied.

Each test of the functionality of proved the specification has been met as they proof regardless of initial input the provided as the values were randomly generated that were used in testing.

Conclusion

The creation of this program allowed for the design and visually communicating to others the structure of the software solution. Use was made of inheritance which allowed for the generalisation of a behaviour shared by a set of classes. Allowing a decrease in the amount of code required to be written. Use made of Arrays of the super class which allow for objects which shard a super class of the type of array to be stored in the single array allowing for the data of both type to be loop over.

Difficulties encounter in the creation of this program when looping over a loop which contains it subtypes if a method of subtype is required to be called. It will return a Compilation error. So type checking was required to check if that item in array was of subtype then it had to be casted to that of the type of subtype before it could be ran.

Given more time a system which uses there previous holidays of a client to determine likely future holidays would be developed. This would allow for uses to get recommended holidays with greater similarity to there preferences.

Use were made of `enums` which allowed the definition of predefined type which allow for variables to only take on the values of set constants inside the `enums`.