

**ASSIGNMENT COVER SHEET**

**Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ID Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Course: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Year: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lecturer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Title of Assignment: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Due Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date Submitted: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The material contained in this assignment is the authors original work, except where work quoted is duly acknowledged in the text. No aspect of this assignment has been previously submitted for assessment in any other unit or course.

Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_

Table of contents

[1. How this application works 3](#_Toc475542422)

[Figure 1.1 MainPage.java 3](#_Toc475542423)

[Figure 1.2 Dublin.java 4](#_Toc475542424)

[Figure 1.3 Dublin.java 5](#_Toc475542425)

[Figure 1.4 Navan.java 5](#_Toc475542426)

[Figure 1.5 Drogheda.java 6](#_Toc475542427)

[Figure 1.6 Tralee.java 6](#_Toc475542428)

[2.Design pattern used to solve assignment 7](#_Toc475542429)

[2.1 Abstract Factory pattern 7](#_Toc475542430)

[Figure 2.1.1 Ireland.java 7](#_Toc475542431)

[Figure 2.1.2 Town.java 8](#_Toc475542432)

[Figure 2.1.3 Hotel.java 8](#_Toc475542433)

[Figure 2.1.3 Dublin.java 9](#_Toc475542434)

[3. UML class diagram for Assignment 1 10](#_Toc475542435)

[4. Extension of the application 11](#_Toc475542436)

[4.1 Creating a new town 11](#_Toc475542437)

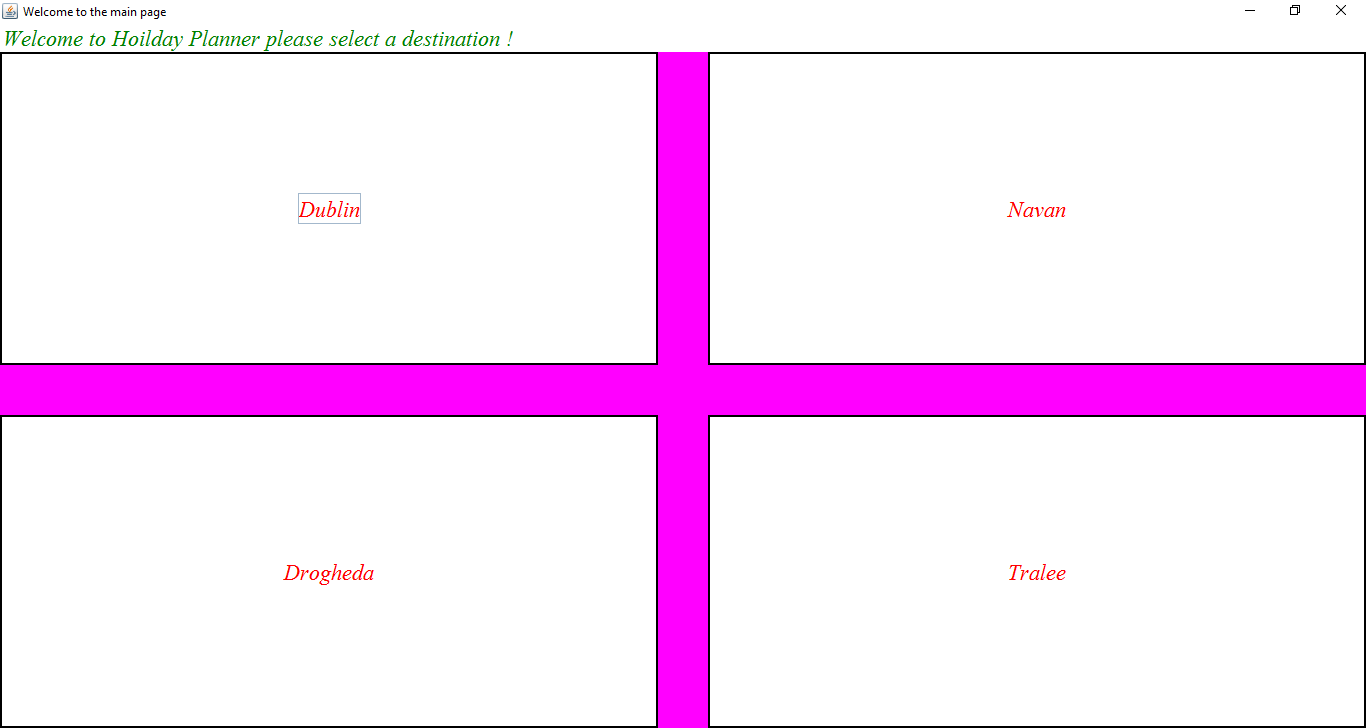
[4.2 Add a sports facility to existing towns 11](#_Toc475542438)

[5.Bibliography 11](#_Toc475542439)

# **1. How this application works**

The main page of the application brings the client to an interface consisting of four JButtons Dublin, Navan, Tralee, and Drogheda each of which brings the user to their respective pages.

## Figure 1.1 MainPage.java



The four pages of the towns have a consistent layout of gridlayout. Each page consists of a different town synopsis and map of how to get to the selected town. The user can then select one of the 4 main radio buttons and find out information on the name, description and location of their selected item (Hotel, Restaurant, places of interest and things to do) by selecting on their respective JButtons under the white panel. The information is then drawn on by a paint method on to this white panel. Selecting another item e.g. restaurant will invoke a clear method that will clear the panel and allow the new selection to be drawn on to the panel.

## Figure 1.2 Dublin.java



When hotel is selected and all the buttons are selected (Name, description and location) the strings will be drawn on to the panel

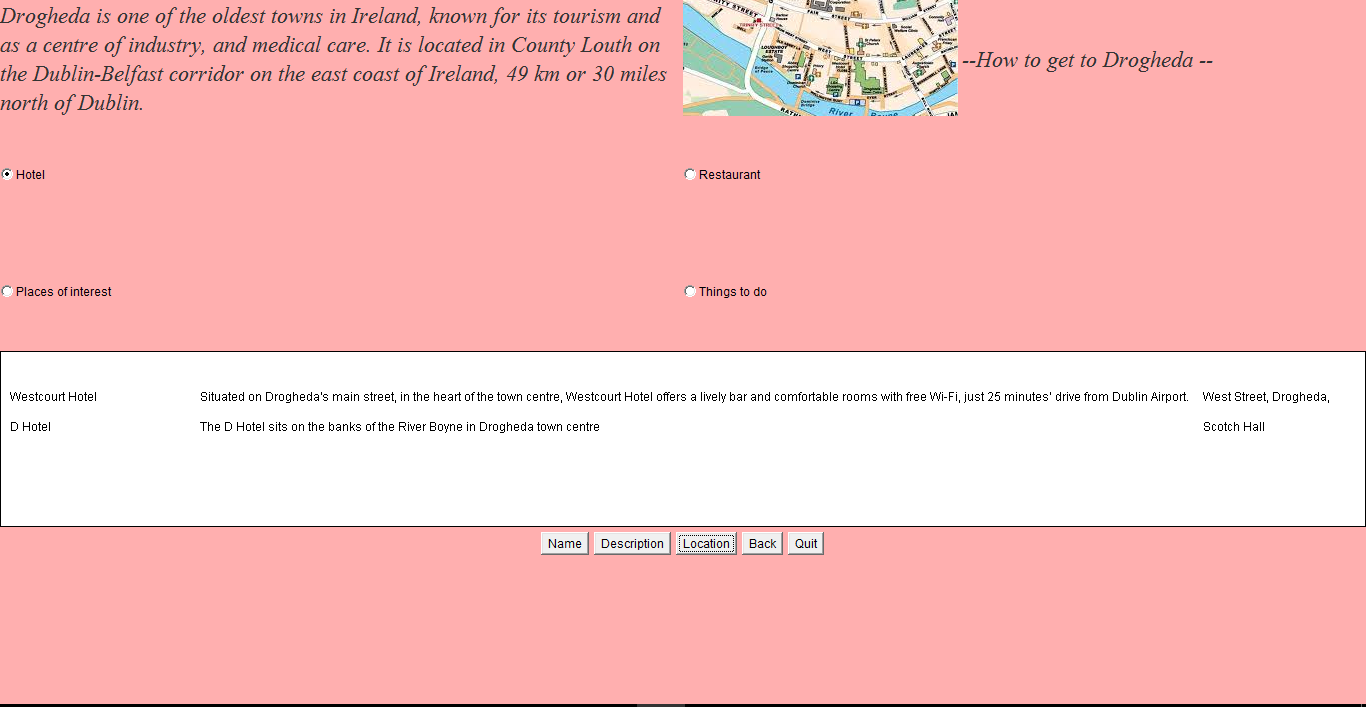
## Figure 1.3 Dublin.java



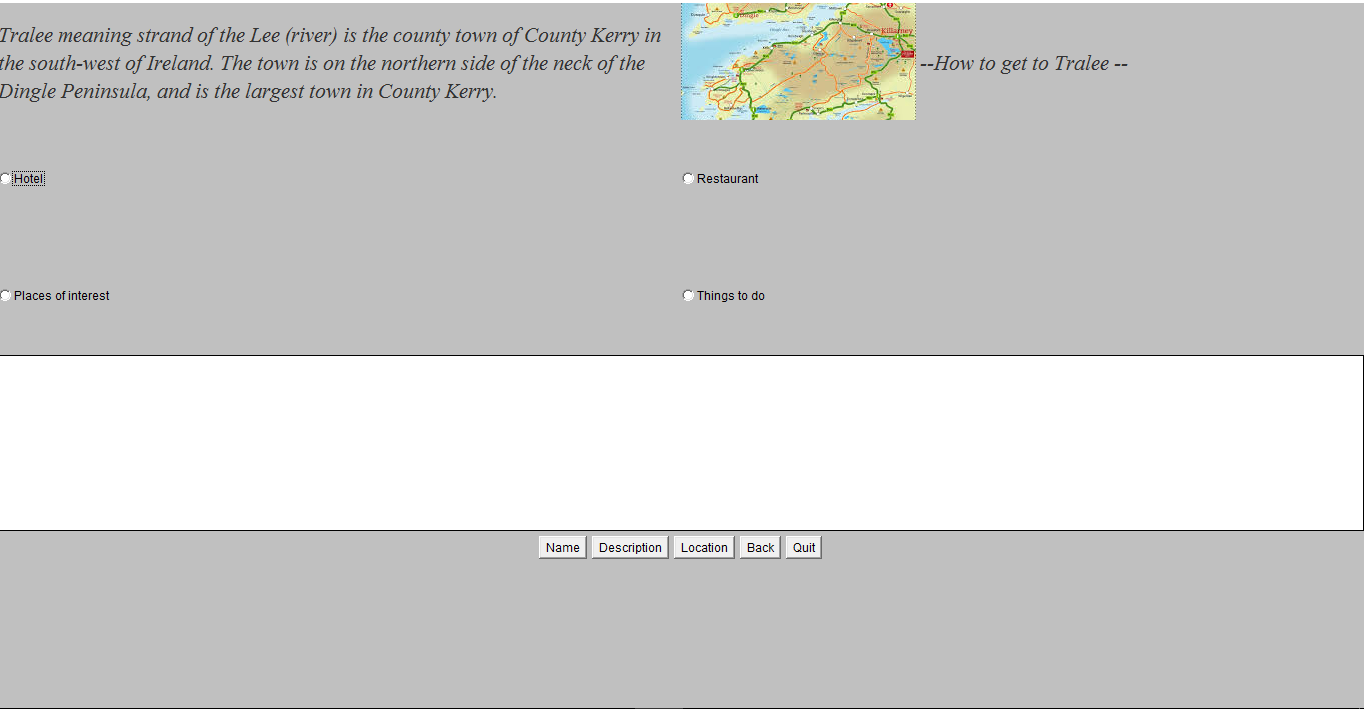
The other pages Navan Tralee and Drogheda work on the exact same principle as the Dublin.java to offer consistency to the user of course the synopsis is different and the information on the hotels and restaurants are but they all use the same methods to extract and display the data their screenshots are as follows.

## Figure 1.4 Navan.java





## Figure 1.5 Drogheda.java



## Figure 1.6 Tralee.java

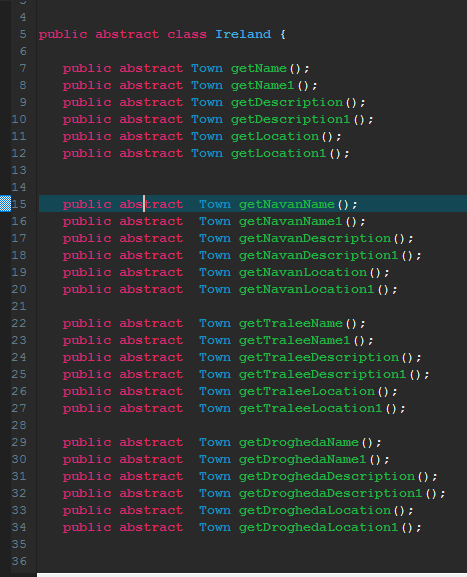
# **2.Design pattern used to solve assignment**

## 2.1 Abstract Factory pattern

The pattern that was used to implement this project was a creational pattern known as the Abstract factory pattern. The reason why this pattern was used say over the simple factory pattern is that you can use the abstract factory pattern to return one of several related classes of objects in this case it was towns of Ireland each of which can return several different objects on request like the hotels restaurants etc.

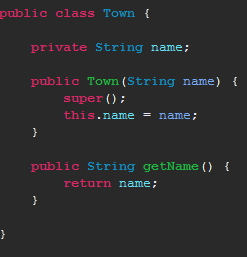
An abstract class was created called Ireland that was used to get each towns Hotel and Restaurant etc. information e.g. Name, description and location. This Ireland class is the abstract factory it defines the methods of one class (Ireland) that can return one of several classes (getName, getName1 and getDescription)

## Figure 2.1.1 Ireland.java



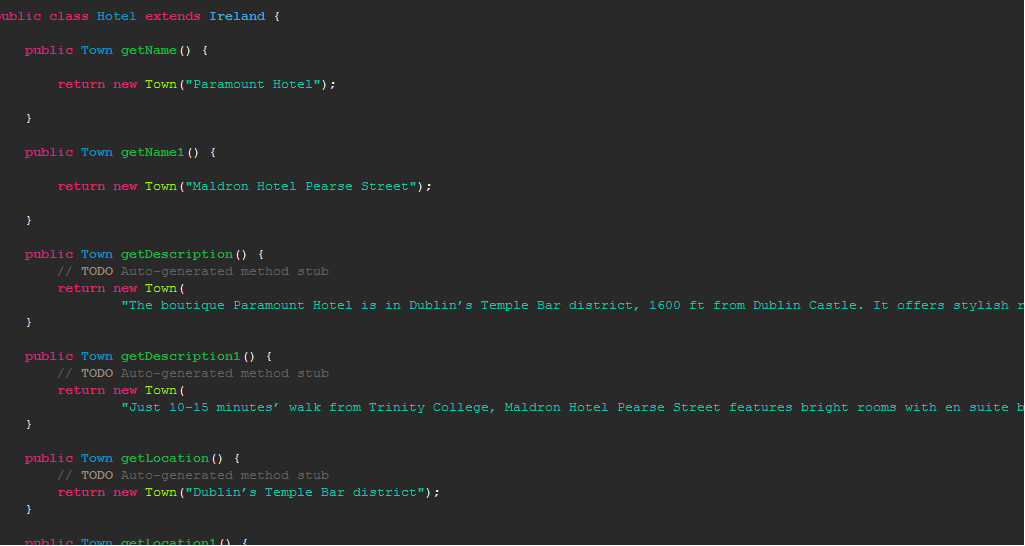
The Town class just returns a string name

## Figure 2.1.2 Town.java



Ireland classes where created for Hotels, Restaurant, Interests and things to do. This is what is known as a concrete factory meaning it implements methods from the parent abstract class (Ireland.java)

## Figure 2.1.3 Hotel.java

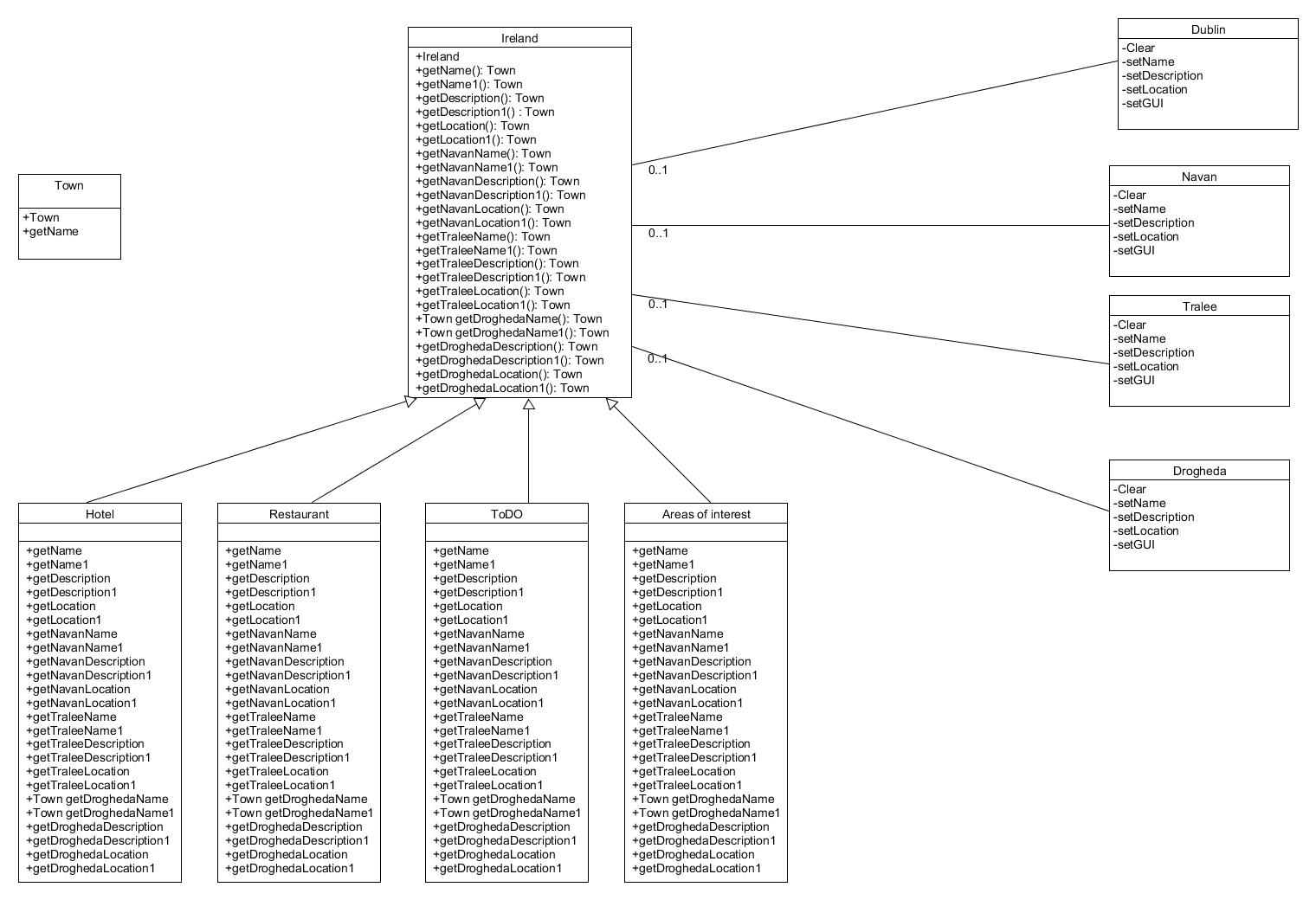


Simply called the methods and repainted the panel these methods are what display on the panel in the main interface of each town.

## Figure 2.1.3 Dublin.java



# 3. **UML class diagram for Assignment 1**



# **4. Extension of the application**

## 4.1 Creating a new town

If one where to extend this application the abstract pattern makes it very easy to do so. Simply copy the interface of one of the towns e.g. Dublin change the name arounds to suit your interests and add the get methods in the abstract class e.g. Public abstract town getWexfordDescription() etc. Implement these methods in the other 4 files (Hotel, Restaurant, Things to do and areas of interest) Use the setWexfordDescription() to set a label or string to return the necessary information on the town.

## 4.2 Add a sports facility to existing towns

To add a sports facility to a existing towns is also very straight forward you would go in to the abstract class Ireland and add getSportsFacilitys() to all the towns. Create a new concrete factory class called sportsFacilitys that extends the Ireland class and implement its getsportsFacilitys() methods and use a set method to sort a string or label to return the information.

# **5.Bibliography**

Abstract Factory .NET Design Pattern in C# and VB - dofactory.com. (2017). [online] Dofactory.com. Available at: http://www.dofactory.com/net/abstract-factory-design-pattern [Accessed 22 Feb. 2017].

Booking.com: The largest selection of hotels, homes, and holiday rentals. (2017). [online] Booking.com. Available at: https://www.booking.com/index.en-gb.html?label=gen173nr-1DCAEoggJCAlhYSDNiBW5vcmVmaGmIAQGYAS64AQzIAQ\_YAQPoAQGSAgF5qAID;sid=e2894141a69b4b193192ca88ea976888;sb\_price\_type=total& [Accessed 22 Feb. 2017].

TripAdvisor: Read Reviews, Compare Prices & Book. (2017). [online] Tripadvisor.ie. Available at: https://www.tripadvisor.ie/ [Accessed 22 Feb. 2017].

UMLet - Free UML Tool for Fast UML Diagrams. (2017). [online] Umlet.com. Available at: http://umlet.com/ [Accessed 22 Feb. 2017].