|  |  |
| --- | --- |
| **Student Name:** | **Thomas Chen** |
| **Student ID:** | **20093171** |
| **Task:** | **3** |
| **Task Title:** | **Portfolio: Task 3** |
| **Task Code:** | **AT2 POR-Task-3** |
| **Due Date:** | ***Session 12***  *Please see blackboard for the most accurate due date.* |
| **Notes:** | *If you did not DOUBLE CLICK this template, then close the file and do so to create your blank report.* |

Table of Contents

[01 Create a new pycharm project 3](#_Toc79483181)

[02 Create an ‘event’ class 4](#_Toc79483182)

[03 Print human readable info 4](#_Toc79483183)

[04 SImple testing and inspection 4](#_Toc79483184)

[05 Events are, well, events 4](#_Toc79483185)

[06 Analysis: Class Diagram 4](#_Toc79483186)

[07 Analysis: adding events 4](#_Toc79483187)

[08 Analysis: Sequence Diagram (success) 4](#_Toc79483188)

[09 Sequence Diagram (alternate scen.) 4](#_Toc79483189)

[10 Implement the method 4](#_Toc79483190)

[11 Keep your project safe Error! Bookmark not defined.](#_Toc79483191)

[Appendix A: Template Instructions 4](#_Toc79483192)

[Appendix B: Before Submission 4](#_Toc79483193)

# 01 Download Smileys Project

#### Provide screenshot of the commands from the assessment tool:





# 02 Find the Classes

#### Answer the following questions:

#### How many classes can you identify in the project? 4 Classes

#### In your own words, describe how ‘abstraction’ is visible in this project. The abstract class Blinkable has a non-defined abstract method (blink). The class Happy, is a sub class of Blinkable and provides an implementation for the blink method.

#### Describe which of these classes are subclasses and which are base (or super) classes.

#### Happy is a sub class of Smiley and Blinkable

Blinkable is a sub class of ABC

Sad is a subclass of Smiley

Finally, Smiley is the super class

#### What is the name of the process of deriving from base classes?

#### Inheritance

# 03 Not all classes are the same

#### In your own words, describe:

#### The commonalities between Happy and Sad Happy and sad are both subclasses of Smiley and they both instantiate their own draw\_mouth and draw\_eyes methods

#### The differences between Happy and Sad Happy is also a sub class of blinkable while sad isn’t. Otherwise they are essentially the same, besides that they light up different spots of the LED lights, to emulate happiness and sadness on a face.

#### Is there anything that stands out as a difference?

Mainly that happy is a subclass of blinkable

# 04 Where’s the sensehat in that?

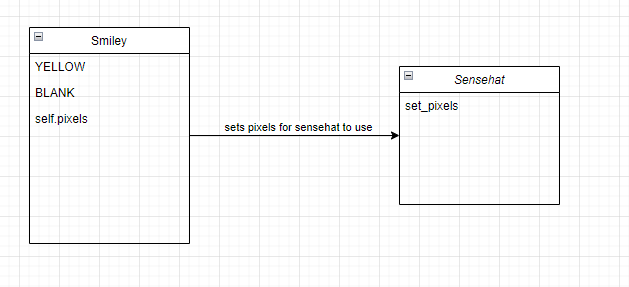
#### Answer the following questions:

#### Where is the SenseHat used (in which **class**)? Sense hat is used in Smiley.py

#### Which functionalities of the SenseHat are used? The Smiley class sets the Sensehat’s display to light up or dim

#### What is the process of storing and potentially hiding objects in classes called? Encapsulation

#### Draw a simple Class Diagram that shows the SenseHat and the containing class (the answer from a.). Make sure to use the correct relationship and multiplicities.



# 05 Sad smileys can’t blink

#### Follow the instructions (and answer any questions):

#### In your own words, how does the **blink()**method make the smiley blink? The wide\_open variable is default set to true. In the Happy class, if wide\_open is true, the pixels for eyes will be blank (eyes are open).

#### In the blink() method, wide\_open is set to false. If wide\_open isn’t true, the pixel for the eyes will be yellow (eyes are closed)

#### Create a new method called blink in the **Sad** class and ensure you use the same prototype (=name + arguments):

#### Implement the code that makes the smiley blink. You may use the implementation from **Happy** as guidance.

#### Test the code on your Raspberry Pi and watch the sad smiley blink its eyes. (You may have to adjust the **main.py** script for this.)

# 06 If it walks like a duck…

#### Answer the following questions:

#### What kind of class is **Blinkable**? Look at its super class for a hint. Blinkable is an abstract base class

#### Any class that uses **Blinkable** is said to “implement” it. What is another (generic) name for a class like **Blinkable**, which may be implemented by other classes? super class

#### What is the answer from the previous question an example of? Choose from: Abstraction, Polymorphism, Inheritance, and Encapsulation. Inheritance

#### Why were you able to add a method called **blink()** to **Sad** and use it exactly like the one defined in **Happy**? Sad and Happy are both inherited from Smiley, therefore they have the same/similar parameters and methods. Therefore, I can add blink() to sad with the same code

#### Regarding the answer to d., how is this generally called and why does it work in Python but not in (some) other languages, like C#?

Multilevel inheritance

Other languages like C# will have issues if there is more than one interpretation of a method

# 07 Does a smiley have to be yellow?

#### Answer the following questions:

#### Which colours are **defined** and where? Colours; white, green, red, yellow and blank (no lights are on) are defined in the Smiley class

#### What is the name of the **type of variables** that hold the colours? Tuple

#### Where are the colour variables actually **used**? In smiley’s initialisation

#### in Happy’s and Sad’s draw\_mouth, draw\_eyes and blink functionj

#### What would be an easy, albeit rather naive, way to change the colour of the smileys, for example, to green?

In the Smiley class, change the Y variable to be self.GREEN

# 08 Flexible colours (1)

#### Follow these instructions:

#### Added an instance method called ***complexion*** to the class ***Smiley***. You can make it return ***self.YELLOW***. (Technically speaking, smileys don’t have a complexion as they have no skin, but it sounds a bit nicer than just “colour”.)

#### Update the subclasses that use the colour variable directly to, instead, use the new method ***complexion***.

#### Which of the four pillars of OO have we used here: Abstraction, Polymorphism, Inheritance, or Encapsulation? (If you feel more than one applies, pick the one that applies **best**.) inheritance

#### Check that the new code works correctly. It should still just show a yellow smiley.

# 09 Flexible colours (2)

#### Follow these instructions:

#### Add a default argument to the magic method ***\_\_init\_\_()*** of the ***Smiley*** class and call it ***complexion***. Assign ***YELLOW*** to it.

#### Create a new **instance variable** called ***my\_complexion*** and assign ***complexion*** to it.

#### Why do we need to call this instance variable ***my\_complexion*** and can’t just use ***complexion***?

#### Also assign ***complexion*** to ***Y***.

#### Finally, update the **method** ***complexion*** and have it return ***self.my\_complexion***.

#### Run the program and see that it still just creates a yellow smiley…

# 10 Flexible colours (3)

#### Follow these instructions:

#### Locate the initialiser method of the class ***Sad*** and within it the call to its super class’s initialiser method.

#### Change the call as follows: ***super().\_\_init\_\_(complexion=self.BLUE)***

#### Run code to show that the sad smiley is now blue.

#### Run code to show that the happy smiley is still yellow.

#### In your own words, **describe** how you would create a new angry smiley that has a Red complexion and angry looking eyes.

For the angry smiley, I’d change the eyes to become a triangle to indicate angry eyes

# Appendix A: Template Instructions

Some basic instructions on using this template.

These instructions include:

* Adding references, and
* What to do **before** submitting the assessment.

# Appendix B: Before Submission

Before you submit the assessment:

1. Ensure you have completed all the steps and answered all questions in the assessment task.
2. Then update the Table of Contents
3. Save this document as per instructions in the assessment task.

Once all the above is completed then you are able to submit the work for assessment.