**Memory V2 Reflection Activity**

**Q1. Describe how the game class determines when to stop the game, and list all the relevant code segments.**

|  |
| --- |
| There is a instance attribute self.continue\_game in the Game object. When this attribute is assigned to False, the game class stop the game.  Code fragments:  def decide\_continue(self):  # Check and remember if the game should continue  # - self is the Game to check  self.continue\_game = False  for row in self.grid:  for tile in row:  if not tile.is\_exposed:  self.continue\_game = True |

**Q2. Assume you also want to keep track in the game of how many times the user clicked anywhere in the window, independent of whether a tile is exposed or not. How do you have to change the code for the game class?**

|  |
| --- |
| 1. add a new attribute self.count to Game class to memorize how many times the user clicked and initialize it as 0, and make the following change in the handle\_event method of the game class   def handle\_event(self):  # handle the event produced by the player input  event = pygame.event.poll()  if event.type == QUIT:  self.close\_clicked = True  if event.type == MOUSEBUTTONDOWN and self.continue\_game is True:  self.handle\_mouseup(event)  self.count += 1 |

**Q3. Write a method in the Tile class that takes a string, denoting a color name such as ‘red’, ‘blue’, etc) as input and changes the border color of the tiles to that color.**

|  |
| --- |
| @classmethod  def change\_br\_color(cls, color):  cls.br\_color = color |

**Q4. Assume that you want to show a ticking “clock” *below* the score of the memory game. For this purpose, you have stored 60 images in an instance attribute of the game class self.clock\_images, which is a list with 60 elements, storing images of a “clock” with a second hand for each of the 60 seconds around the clock. The images should be shown in this sequence, changing the image that is drawn every second, over and over, starting with the 0th second, then the 1st second, and so on up to the 60th second, and then starting at be beginning again. Extend the draw\_score method accordingly (no change is necessary anywhere else in the code).**

|  |
| --- |
| add the following lines after the current code in draw\_score:  img = self.clock\_images[clock%60]  draw(img) |

**Q5. Consider the implementation of the select method in the Tile class. Re-write the method suite so that it consists of only a single return statement!**

|  |
| --- |
| def select(self, position):  # handle the click action  # position passed is event.pos, indicaing the osition of the click  return self.doselect(position)  def do\_select\_thing(self, position):  if self.rect.collidepoint(position):  if not self.is\_exposed:  self.is\_exposed = True  return True  return False |