CSCI 145 Quiz 10 practice questions on inheritance and games **ANSWER KEY**

Special methods, inheritance, and polymorphism (Lecture 19)

1. The class `Weight` is below. These lines run in the Shell, in order. Write what the Shell shows for each of the last four lines.

```
class Weight:
    def __init__(self, grams):
        self.grams = grams
    def __str__(self):
        return str(self.grams) + " g"
    def __eq__(self, other):
        return self.grams == other.grams
    def __add__(self, other):
        return Weight(self.grams + other.grams)
```

```
>>> a = Weight(250)
>>> b = Weight(250)
```

```
>>> print(a)          250 g
>>> a == b            True
>>> a is b            False
>>> print(a + b)      500 g
```

Questions 2 to 4 use the classes `Boat`, `Sailboat`, and `Canoe` on the left. Questions 2 and 3 use the code on the right.

```
class Boat:
    def __init__(self, name, length):
        self.name = name
        self.length = length
    def fee(self):
        return 2 * self.length
    def kind(self):
        return "boat"
    def __str__(self):
        return self.name + " is a " + self.kind()

class Sailboat(Boat):
    def __init__(self, name, length, sails):
        super().__init__(name, length)
        self.sails = sails
    def kind(self):
        return "sailboat"

class Canoe(Boat):
    def fee(self):
        return 5
    def kind(self):
        return "canoe"
```

```
s = Sailboat("Wren", 20, 2)
c = Canoe("Otter", 16)
b = Boat("Tug", 30)
for boat in [c, b, s]:
    print(boat.kind(), boat.fee())
print(s)
```

2. Whose method runs: `Boat`'s, `Sailboat`'s, or `Canoe`'s?

making c runs `__init__` **Boat's**

`s.fee()` **Boat's**

`c.fee()` **Canoe's**

`print(s)` runs `__str__` **Boat's**

then it calls `self.kind()` **Sailboat's**

3. What does the code print?

Output
```
canoe 5
boat 60
sailboat 40
Wren is a sailboat
```

4. Write a class `Kayak`, a child class of `Boat`. Do not include docstrings.
   - `Kayak("Minnow", 12, 2)` makes a kayak with the name `"Minnow"`, the length 12, and a new attribute `seats` that is 2. Use `super().__init__`.
   - Its `kind` method returns `"kayak"`.

   For example:

```
>>> k = Kayak("Minnow", 12, 2)
>>> print(k)
Minnow is a kayak
>>> k.seats
2
>>> k.fee()
24
```

```
class Kayak(Boat):
    def __init__(self, name, length, seats):
        super().__init__(name, length)
        self.seats = seats
    def kind(self):
        return "kayak"
```

Animation and games (Lecture 20)

On this page, the screen is 600 pixels wide and 400 pixels tall: `SCREEN_WIDTH` is `600` and `SCREEN_HEIGHT` is `400`.

5. This is the game loop of a program with one ball. Answer each question with a line number.

```
 1  running = True
 2  while running:
 3      for event in pygame.event.get():
 4          if event.type == pygame.QUIT:
 5              running = False
 6      ball.update()
 7      screen.fill(BG_COLOR)
 8      ball.draw(screen)
 9      pygame.display.flip()
10      clock.tick(60)
```

a) Which line paints over the last frame? **7**

b) Which line puts the finished frame on the screen? **9**

c) Which line keeps the loop to at most 60 frames a second? **10**

d) When the player closes the window, which line changes `running` so the loop ends? **5**

6. These two lines draw a rectangle and a circle.

```
pygame.draw.rect(screen, (0, 200, 0), (100, 250, 80, 40))
pygame.draw.circle(screen, (255, 255, 0), (450, 100), 30)
```

a) The rectangle's upper-left corner **(100, 250)**

b) The rectangle's lower-right corner **(180, 290)**

c) The y of the top of the circle **70**

d) Which shape is higher up? **the circle**

7. This is the part of `Ball.update` that changes `x` and `vx`.

```
def update(self):
    self.x = self.x + self.vx
    if self.x - self.radius < 0 or self.x + self.radius > SCREEN_WIDTH:
        self.vx = -self.vx
```

A ball has `x = 585`, `radius = 10`, and `vx = 4`. Then `ball.update()` is called three times. Write `x` and `vx` after each call.

after	x	vx
1 call	**589**	**4**
2 calls	**593**	**-4**
3 calls	**589**	**-4**

8. `PLAYER_SIZE` is `40`. `move` keeps the player on the screen, as in the videos.

a) A player has `rect.x = 550` and `rect.y = 10`. Then `player.move(20, -20)` runs. What are `rect.x` and `rect.y` now? **560 and 0**

b) A rock has `x = 590`, `y = 30`, and `radius = 20`. What does `rock.get_rect()` return? **Rect(570, 10, 40, 40)**

c) Another player's `rect` is `Rect(300, 360, 40, 40)`. Another rock's `get_rect()` returns `Rect(330, 330, 40, 40)`. Is `player.rect.colliderect(rock.get_rect())` `True` or `False`? **True**

9. A `Bubble` has the attributes `x`, `y`, `radius`, and `speed`. Bubbles float up the screen. Write the `update` method of the class `Bubble`. Do not include a docstring.

- It moves the bubble up by `speed`.
- Then, if the bubble is completely above the top of the screen (`y + radius` is less than 0), it sets `y` to `SCREEN_HEIGHT + radius`. Otherwise, nothing else changes.

```
def update(self):
    self.y = self.y - self.speed
    if self.y + self.radius < 0:
        self.y = SCREEN_HEIGHT + self.radius
```