📖 **task2.md**

PreRelease Task2

General Marks:

- 4 marks for ontime (submitted 3/25/2018 or earlier)
- 1 mark for being in a "prerelease/task2.md" file

category	marks
general	[5 marks]
2.1	[29 marks]
2.2	[2 marks]
2.3	[4 marks]
2.4	[14 marks]
2.5	[9 marks]
2.6	[8 marks]
2.7	[9 marks]
2.8	[3 marks]
2.10	[5 marks]
total:	[88 marks]

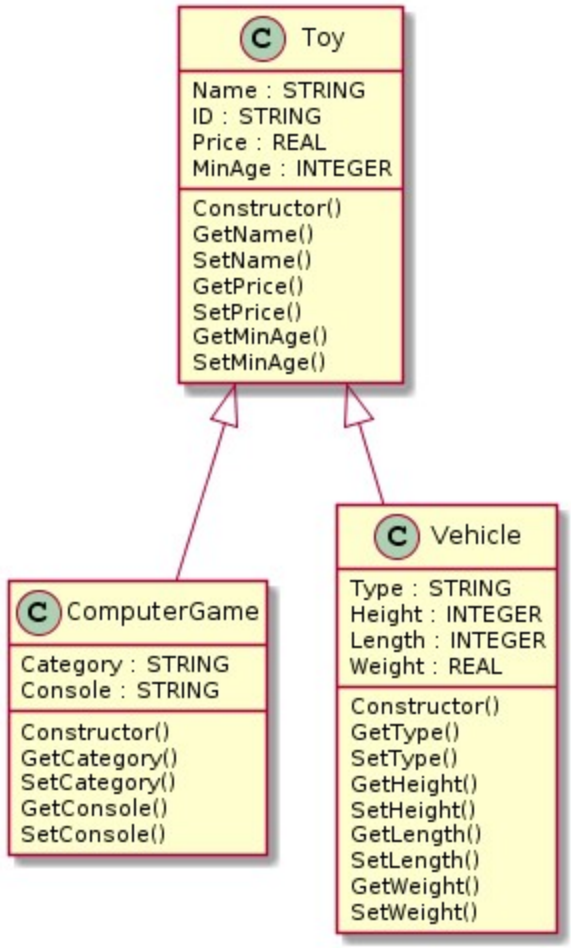
2.1 - 2.2 [29 marks]

marks:

- 10 marks for attributes
- 10 marks for attribute types
- 3 marks for constructors
- 6 marks for at least 1 Getter and Setter in each class

2.2 [2 marks]

- 2 marks for arrows



2.3 [4 marks]

marks

- constructor
- private attributes
- at least one getter
- at least one setter

```
class Toy:
    def __init__(self,name,id,price,minAge):
        self._name = name
        self._id = id
        self._price = price
        self._minAge = minAge

    def getID(self):
        return self._id

    def setID(self, id):
        self._id = id

    def getName(self):
        return self._name

    def setName(self, name):
        self._name = name
```

```
    def getPrice(self):
        return self._price

    def setPrice(self,price):
        self._price = price

    def getMinAge(self):
        return self._minAge

    def setMinAge(self, minAge):
        self._minAge = minAge
```

Task 2.4 [14 marks]

For each class:

- 1 mark for inheriting from Toy class
- 1 mark for constructor
- 1 mark for constructor calling Toy constructor
- 1 mark for private attributes
- 1 mark for general syntax
- 1 mark for having a getter function
- 1 mark for having a setter function

ComputerGame Class

```
class ComputerGame(Toy):
    def __init__(self, name, id, price, minAge, category, console):
        Toy.__init__(self, name, id, price, minAge)
        self._category = category
        self._console = console

    def getCategory(self):
        return self._category

    def setCategory(self, category):
        self._category = category

    def getConsole(self):
        return self._console

    def setConsole(self, console):
        self._console = console
```

Vehicle class

```
class Vehicle(Toy):
    def __init__(self, name, id, price, minAge, type, height, length, weight):
        Toy.__init__(self, name, id, price, minAge)
        self._type = type
        self._height = height
        self._length = length
        self._weight = weight

    def getType(self):
        return self._type

    def getHeight(self):
```

```
        return self._height

    def getLength(self):
        return self._length

    def getWeight(self):
        return self._weight

    def setType(self, type):
        self._type = type

    def setHeight(self, height):
        self._height = height

    def setLength(self, length):
        self._length = length

    def setWeight(self, weight):
        self._weight = weight
```

Task 2.5 [9 marks]

Note:

- I have changed the constructor code to call the setter functions, this is not manditory, but enables code reuse

Marks / class:

- at least one check in a setter of each class
- at least one check in a constructor of each class
- general syntax

Toy Class (others not shown)

```
class Toy:
    def __init__(self,name,id,price,minAge):
        Toy.setName(self,name)
        Toy.setID(self, id)
        Toy.setPrice(self,price)
        Toy.setMinAge(self,minAge)

    def setName(self, name):
        if not isinstance(name, str):
            raise ValueError("name must be string")
        if len(name) < 1:
            raise ValueError("invalid name", name)
        self._name = name

    def setID(self, id):
        if not isinstance(id, str):
            raise ValueError("id must be string")
        if len(name) <> 8:
            raise ValueError("id numbers are 8 characters", id)
        self._id = id

    def setPrice(self,price):
        if price < 0:
            raise ValueError("price must be positive")
```

```
        self._price = price

    def setMinAge(self, minAge):
        if minAge < 0:
            raise ValueError("minAge must be positive")
        if minAge > 18:
            raise ValueError("minAge too old")
        self._minAge = minAge

...
```

2.6 [8 marks]

marks / Call:

- 1 mark for general syntax
- 1 mark for creating an instance
- 1 mark for correct parameters
- 1 mark for adding to array

```
rsc = Vehicle("Red Sports Car", "RSC13", 15.0, 6, "Car", 3.3, 12.1, 0.08)
l4d = ComputerGame("l4d", "L4D01", 10, 15, "FPS", "PC")

toys = []
toys.append(rsc)
toys.append(l4d)
```

2.7 [9 marks]

marks:

- general syntax [1 mark]
- having a findToy procedure [1 mark]
- handling the case where the id is not found [1 mark]
- toString() OR printDetails() added to all three classes [6 marks]

Toy class:

```
def __str__(self):
        return "{} [{}] \n Price: ${:.2f} \n MinAge: {}".format(
            self._name,
            self._id,
            self._price,
            self._minAge)
```

ComputerGame overrides this:

```
    def __str__(self):
        return "{}\n {} \n {} game".format(
            Toy.__str__(self),
            self._category,
            self._console)
```

Vehicle also overrides this:

```
    def __str__(self):
        return "{}\n {} [{} x {} x {}]".format(
            Toy.__str__(self),
            self._type,
            self._height,
            self._length,
            self._weight)
```

findToy procedure

```
def findToy():
    id = input("Enter the toy ID:")
    for t in toys:
        if (t.getID() == id):
            print(t)
            exit()
    print("Toy with id `" + id + "` not found")
```

testing with appropriate data

```
Enter the toy ID:RSC13
Red Sports Car [RSC13]
 Price: $15.00
 MinAge: 6
 Car [3.3 x 12.1 x 0.08]


Enter the toy ID:L4D01
l4d [L4D01]
 Price: $10.00
 MinAge: 15
 FPS
 PC game


 Enter the toy ID:xxxxxx
Toy with id `xxxxxx` not found
```

2.8 [3 marks]

marks:

- 1 mark added discount() to Toy class
- 1 mark general syntax
- 1 mark correct math

Toy class

```
    def discount(self, amount):
        d = (100-amount)/ 100
        self.setPrice(self._price * d)
```

2.10 [5 marks]

- 3 marks for writing one of the algorithms
- 1 mark for using getPrice() rather than element value directly
- 1 mark for general syntax

bubble sort

```
def bubbleSortToys(arr):
    for i in range(len(arr)):
        for j in range(len(arr) - i - 1):
            if arr[j].getPrice() > arr[j + 1].getPrice():
                temp = arr[j]
                arr[j] = arr[j + 1]
                arr[j + 1] = temp
```

insertion sort

```
def insertionSortToys(arr):
    N = len(arr)
    for i in range(1, N):
        j = i - 1
        temp = arr[i]
        while j >= 0 and temp.getPrice() < arr[j].getPrice():
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = temp
```