Session 7: Python Classes and Inheritance

Q.1 [3 mks] Write down and test the following example in your Python code editor.

```
class Animal(object):
    """ Animal abstract data type """
    def __init__(self, age):
        self.age = age
        self.name = None
    def get_age(self):
        return self.age
    def get_name(self):
        return self.name
    def set_age(self, newage):
        self.age = newage
    def set_name(self, newname=""):
        self.name = newname
    def __str__(self):
        return "animal:"+str(self.name)+":"+str(self.age)

# Test code
print("\n---- animal tests ----")
a = Animal(4)
print(a)
print(a.get_age())
a.set_name("fluffy")
print(a)
a.set_name()
print(a)
#a.speak() # error because there is no speak method for Animal class

class Cat(Animal):
    """   Inheritance example  """
    def __init__(self, age, name = None):
        Animal.__init__(self, age)
        self.name = name
    def speak(self):
        print("meow")
    def __str__(self):
        return "cat:"+str(self.name)+":"+str(self.age)

# Test code
print("\n---- cat tests ----")
c = Cat(5)
c.set_name("fluffy")
print(c)
c.speak()
print(c.get_age())

class Fish(Animal):
    """   Inheritance example  """
    def __init__(self, age, fishType):
        Animal.__init__(self, age)
        self.type = fishType
    def __str__(self):
        return "Fish:"+str(self.type)+":"+str(self.age)

# Test code
print("\n---- Fist tests ----")
#  Write your own test code
# types: Clownfish, Shark, Goldfish
################################################################################
```

Q.2 [2 mks] Write down and test the following example in your Python code editor.

```
import random
class Person(object):
    """  Inheritance example  """
    def __init__(self, name, age):
        self.age = age
        self.set_name(name)
        self.friends = []
    def set_name(self, name):
        # write code here
    def get_name(self):
        # write code here
    def get_age(self):
        # write code here
    def get_friends(self):
        # write code here
    def add_friend(self, fname):
        # write code here)
    def age_diff(self, other):
        # write code here)
    def __str__(self):
        return "person:"+str(self.name)+":"+str(self.age)

# Test code
print("\n---- person tests ----")
p1 = Person("Ali", 30)
p2 = Person("Ahmed", 25)
print(p1.get_name())
print(p1.get_age())
print(p2.get_name())
print(p2.get_age())
print(p1)
p1.age_diff(p2)

class Student(Person):
    """  Inheritance example
          a class variable, tag, shared across all instances
    """
    count = 0
    def __init__(self, name, age, major=None):
        Person.__init__(self, name, age)
        self.major = major
        Student.count +=1
        self.Id  = Student.count
    def __str__(self):
        # write code here, to print name, age, Id
    def change_major(self, major):
        # write code here
    def speak(self):
        r = random.random()
        if r < 0.25:
            print("i have homework")
        elif 0.25 <= r < 0.5:
            print("i need sleep")
        elif 0.5 <= r < 0.75:
            print("i should eat")
        else:
            print("i am watching tv")
```

```python
# Test code
print("\n---- student tests ----")
s1 = Student('Ali', 20, "CS")
s2 = Student('Ahmed', 18)
print(s1)
print(s2)
print(s1.get_name(),"says:", end=" ")
s1.speak()
print(s2.get_name(),"says:", end=" ")
s2.speak()

print(Student.count)     # what will happen in this case
print(Student.Id)        # what will happen in this case

################################################################################
```

Q.3 [2 mks] Write down and test the following example in your Python code editor.

```python
class Rabbit(object):
    """   Use of class variables  """
    # a class variable, tag, shared across all instances
    tag = 0
    def __init__(self, age, parent1=None, parent2=None):
        self.parent1 = parent1
        self.parent2 = parent2
        Rabbit.tag += 1
        self.rid = Rabbit.tag
    def get_rid(self):
        # zfill used to add leading zeroes 001 instead of 1
        return str(self.rid).zfill(3)
    def get_parent1(self):
        # write code here
    def get_parent2(self):
        # write code here
    def __add__(self, other):
        # returning object of same type as this class
        return Rabbit(0, self, other)
    def __eq__(self, other):
        # compare the ids of self and other's parents
        # don't care about the order of the parents
        # the backslash tells python I want to break up my line
        parents_same = self.parent1.rid == other.parent1.rid \
                       and self.parent2.rid == other.parent2.rid
        parents_opposite = self.parent2.rid == other.parent1.rid \
                           and self.parent1.rid == other.parent2.rid
        return parents_same or parents_opposite
    def __str__(self):
        return "rabbit:"+ self.get_rid()

# Test code

print("\n---- rabbit tests ----")
print("---- testing creating rabbits ----")
r1 = Rabbit(3)
r2 = Rabbit(4)
r3 = Rabbit(5)
print("r1:", r1)
print("r2:", r2)
print("r3:", r3)
print("r1 parent1:", r1.get_parent1())
print("r1 parent2:", r1.get_parent2())

print("---- testing rabbit addition ----")
r4 = r1+r2   # r1.__add__(r2)
print("r1:", r1)
print("r2:", r2)
print("r4:", r4)
print("r4 parent1:", r4.get_parent1())
print("r4 parent2:", r4.get_parent2())

print("---- testing rabbit equality ----")
r5 = r3+r4
r6 = r4+r3
print("r3:", r3)
print("r4:", r4)
print("r5:", r5)
print("r6:", r6)
print("r5 parent1:", r5.get_parent1())
print("r5 parent2:", r5.get_parent2())
print("r6 parent1:", r6.get_parent1())
print("r6 parent2:", r6.get_parent2())
print("r5 and r6 have same parents?", r5 == r6)
print("r4 and r6 have same parents?", r4 == r6)
```