Omtentamen

Data- och informationsvetenskap:
Objektorienterad programmering och modellering (DA361A)
Teknik och samhälle, Malmö högskola

Januari 2016

Hjälpmedel

Penna och papper.

Anvisningar

Tentamen omfattar 15 uppgifter. Maxpoäng är 35 varpå 18 poäng är gränsen för Godkänt och 26 poäng för Väl Godkänt.

Behandla endast uppgifter på ena sidan av varje blad och markera varje sida med dina initialer (det är tillåtet med fler uppgifter på blad). Skriv läsligt och kommentera utförligt vad du gör, det kan ge dig poäng även om resultatet är fel.

Glöm inte att fylla i försättsbladet, inklusive att kryssa i vilka uppgifter du svarat på.

Lycka till!

Uppgifter

1. Utifrån kodexemplet nedan, vad är en **instans** respektive **klass**? Ni kan hänvisa till del A och B. Beskriv även hur ni kan avläsa denna skillnaden från exemplet (2p)

```
# Del A
class Person(object):
    def __init__(self, name, age):
        self.name = name
        self.age = age

# Del B
p = Person("Sherlock Holmes", 33)
```

2. I kodexemplet nedan finns ett och samma fel i samtliga metoder. Vad är det för fel och varför är detta fel? (2p)

```
class Rectangle(object):
    def __init__(length, width):
        length = length
        width = width

    def set_length(length):
        length = length

    def get_length():
        return length

    def set_width(width):
        width = width

    def get_width():
        return width

    def calculate_area():
        return length * width
```

3. Skapa ett kort kodexempel där ni visar hur metoden `__str__` fungerar. Beskriv sedan varför denna kan anses vara användbar (2p)

4. Inom objektorienterad programmering med Python anses det vara bättre att använda metoder för att hämta (`get`) och ange (`set`) attribut istället för att direkt använda sig av instansens attribut (t.ex. `person.name`). Varför anses det vara bättre och vilken term används för att beskriva detta koncept? (2p)

5. Vid utformandet av ett klassdiagram brukar vi i samband med att vi anger en relation även ange något i stil med `1..*`, `1..1`, osv. Varför gör vi detta och vilken term används för att beskriva detta koncept? (2p)

6. Nämn tre fördelar med att dela in flera klasser genom relationstypen **Arv** (3p)

7. Om en metod i en subklass har samma namn som i sin superklass, vad händer och vad bör vi tänka på för att undvika detta? (2p)

8. I kodexemplet nedan finner ni tre klasser som tillsammans har olika relationstyper, vilka är dessa? (2p)

```
class Animal(object):
    def __init__(self, name, type):
        self.name = name
        self.type = type

class Dog(Animal):
    def __init__(self, name):
        Animal.__init__(self, name, "Dog")

class Kennel(Object):
    def __init__(self, dogs=[]):
        self.dogs = dogs

    def add_dog(self, dog):
        self.dogs.append(dog)
```

9. Skapa ett komplett klassdiagram från kodexemplet nedan. Tänk på att fylla i objekt, relationer, antalsförhållande och vilka värden attribut och metoder har (eller returnerar) (3p)

```
class Card(object):
    '''
    Represents a card of a suit (Spades, Hearts, etc.)
    and rank (Ace, 2, 3, 4, etc.).
    '''
    def __init__(self, suit=0, rank=2):
        self.suit = suit
        self.rank = rank

class Deck(object):
    '''
    Represents a deck of 52 cards.
    '''
    def __init__(self, cards=[]):
        if len(cards) > 0 and len(cards) < 53:
            self.cards = cards

    def shuffle(self):
        '''
        Shuffles and returns the whole deck
        '''
        pass

    def pop_card(self):
        '''
        Removes a card from the deck and returns it
        '''
        pass

    def add_card(self, card):
        '''
        Takes a new card and inserts it into the deck,
        then returns the newly inserted card
        '''
        pass
```

10. Nämn två stycken karakteristiska drag för objektorienterad programmering (2p)

11. Nämn två stycken karakteristiska drag för Unified Modeling Language (UML) (2p)

12. Vilka praktiska fördelar får vi när vi skriver vår kod genom att först skapa ett klass- och sekvensdiagram? (3p)

13. Visa med ett kort kodexempel eller beskriv med ord för vad termen **polymorphism** har för innebörd inom objektorienterad programmering (2p)

14. Vad kan ni utläsa från klassdiagrammet i **Figure 1**? Tänk på innehåll av varje klass, relationer och antalsförhållande. Kan ni dra någon slutsats om vad programmet ska utföra? (3p)

15. Vad kan ni utläsa från sekvensdiagrammet i **Figure 2**? Tänk på vilka aktörer vi har, ordningen och vilka metoder som används. Kan ni dra någon slutsats om vad programmet ska utföra? (3p)

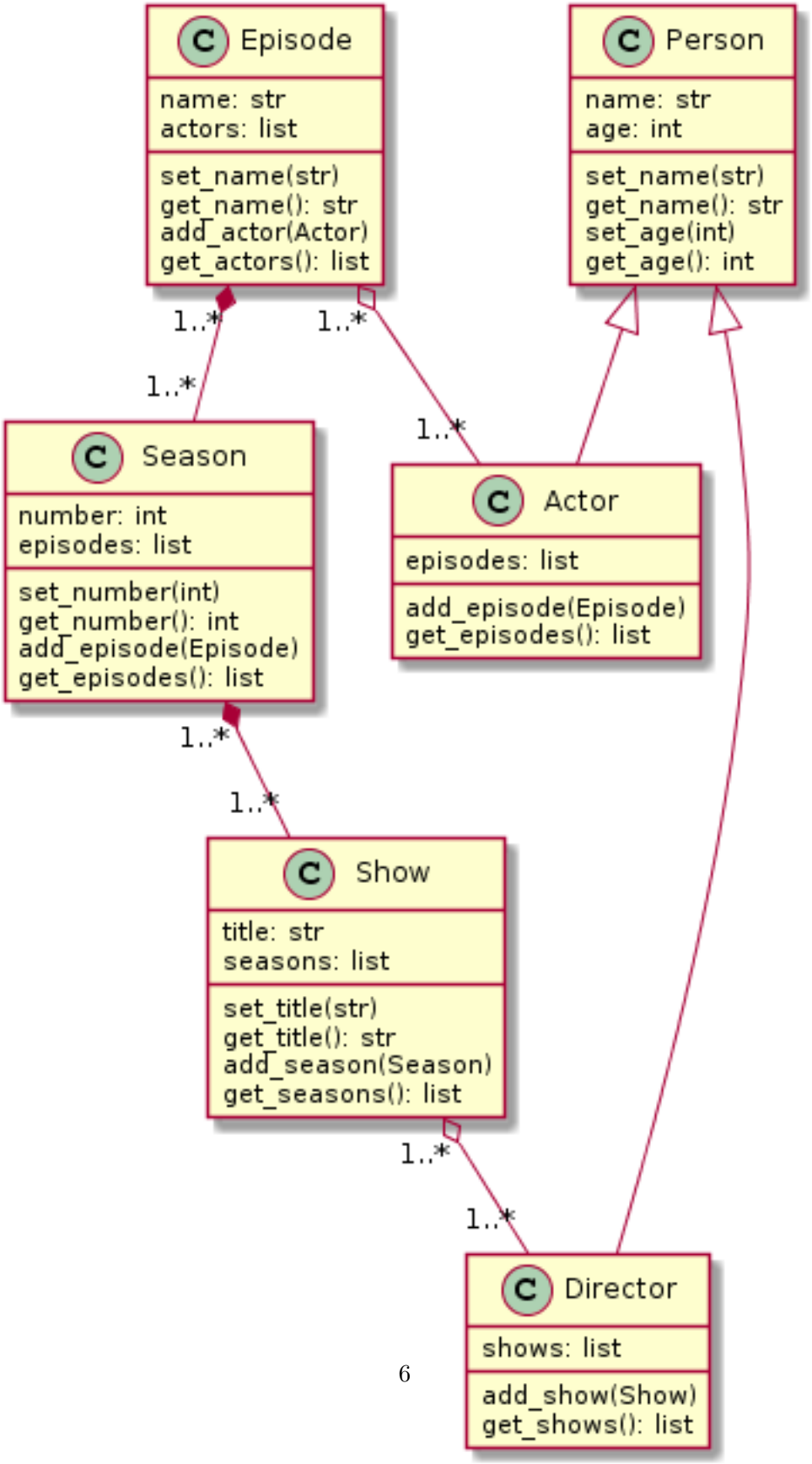

Figure 1: Klassdiagram för uppgift 14

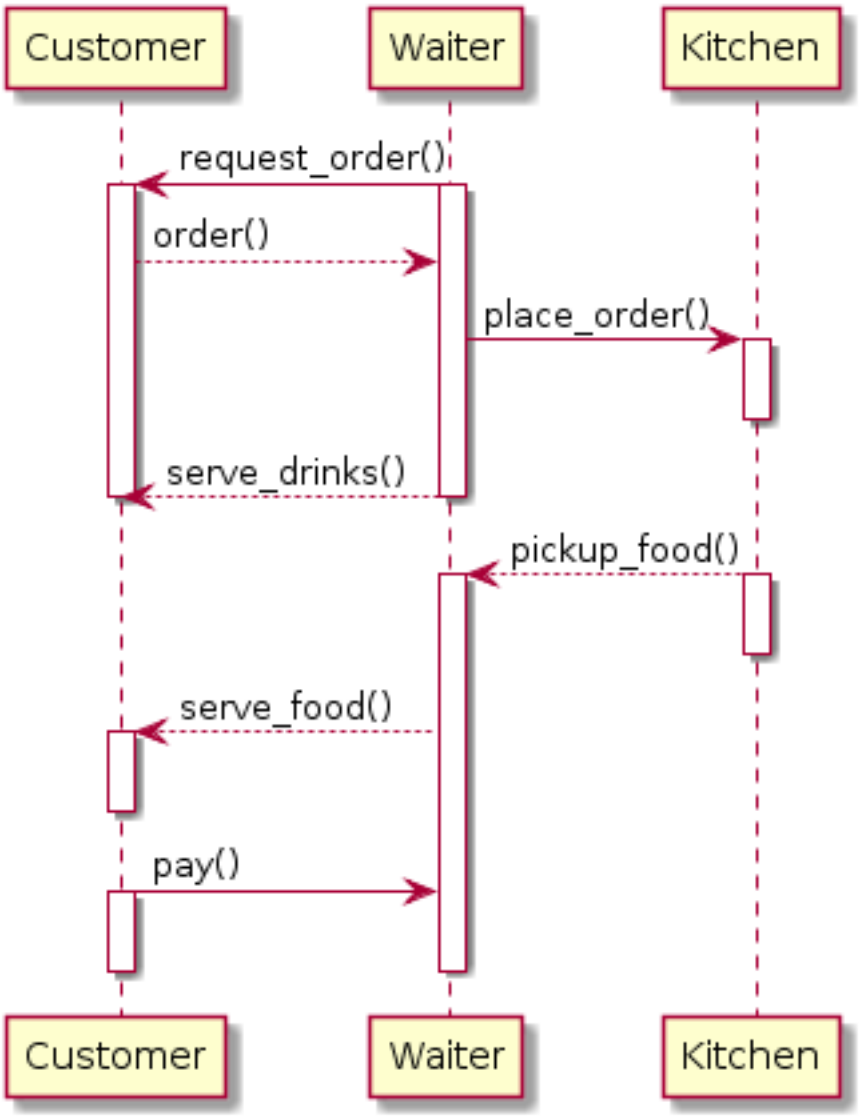

Figure 2: Sekvensdiagram för uppgift 15