Tentamen

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

December 2015

Hjälpmedel

Penna och papper.

Anvisningar

Tentamen omfattar 16 uppgifter. Maxpoäng är 36 varpå 19 poäng är gränsen för Godkänt och 27 poäng för Väl Godkänt.

Behandla endast uppgifter på en sida av varje blad och markera varje sida med dina initialer. 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. Beskriv skillnaden mellan termerna “klass” och “instans” (1p)
2. I definitionen av en Pythonklass används termen “self” som ett argument i samtliga metoder, vad har denna för innebörd? (1p)
3. Utifrån kodexemplet nedan, vad får vi för utskrift och varför får vi denna? (2p)

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

    def sleep(self):
        return "%s is sleeping..." % (self.name)

class Dog(Animal):
    def __init__(self, name, breed):
        # Automatically define the type of animal as "Dog"
        Animal.__init__(self, "Dog", name)
        self.breed = breed

    def sleep(self):
      return "%s the %s, is fast asleep..." % (self.name, self.breed)

class Pug(Dog):
    def __init__(self, name):
        # Automatically define the breed as "Pug"
        Dog.__init__(self, name, "Pug")

    def pug_sleep(self):
        return "*silence* zzZzzZz"

# Instance of "Pug"
douglas = Pug("Douglas")
# Call the "sleep" method
print douglas.sleep()
```

4. Nedanstående kodexempel ger oss en utskrift i stil med `<__main__.Person object at 0x102e36ad0>`, varför? Vad behöver vi komplettera med för att få en utskrift liknande `Jane Doe (25 years)` istället? (2p)

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

jane = Person("Jane Doe", 25)
print jane
```

5. Inom objektorienterad programmering med Python anses det vara dålig praxis att direkt använda sig av eller ändra på attributen för en klass. Givet kodexemplet nedan, vad kan ni ändra på för att förbättra detta? Beskriv vad ni ändrade på och varför, samt vilket samband detta har till termen "inkapsling". (3p)

```
class Movie(object):
        def __init__(self, title, year):
                self.title = title
                self.year = year

        def __str__(self):
                return "%s (%s)" % (self.title, self.year)

movie = Movie("Lord of the flies", 1990)
print movie
# Ger utskriften "Lord of the flies (1990)"

movie.title = "Batman"
movie.year = 1989
print movie
# Ger utskriften "Batman (1989)"
```

6. Utifrån den givna subklassen och exempelutskriften från Pythontolken - vad vet du om superklassen? Beskriv superklassen så detaljerat du kan, använd gärna ett klassdiagram som hjälp för att definiera innehållet. (4p)

   Subklass:

```
class Student(Person):
        """A special type of Person that studies at a university."""
         def __init__(self, name, uni_name):
                 Person.__init__(self, name)
                 self.uni_name = uni_name

         def get_uni_name(self):
                 return self.uni_name

         def __str__(self):
                 return "A student named " + Person.__str__(self)
```

   Exempelutskrift:

```
>>> s = Student("Jane", "Mah")
>>> s.get_uni_name()
'Mah'
>>> s.get_name()
'Jane'
>>> s.say_hello()
Hello, my name is Jane
>>> s.set_name("John")
>>> s.say_hello()
Hello, my name is John
>>> print s
A student named John
```

7. Vad innebär termen kardinalitet i ett klassdiagram? (1p)

8. Beskriv med ett exemepel två typer av relationer som kan användas i ett klassdiagram (metoder behöver ej fyllas i) (2p)

9. Beskriv med ord eller visa med ett kodexempel för vad termen “polymorphism” har för innebörd inom objektorientering. (2p)

10. Diskutera kort kring för- och nackdelar med att skapa följande diagram innan implementationen av ett program: (4p)

    a. Klassdiagram
    b. Sekvensdiagram

11. Skapa ett kodexempel innehållande en implementation av relationstypen aggregat eller komposition mellan två klasser i Python (3p)

12. Vilket eller vilka av följande påstående beskriver objektorientering bäst? (1p)

    1. Objektorientering handlar om att placera variabler och funktioner inom en klassdefinition för att dela upp ett program
    2. Objektorientering krävs för att vi ska kunna strukturera ett program för att det ska kunna köras i en Pythontolk
    3. Objektorientering är ett förhållningssätt för att modellera verkligheten genom att definiera klasser som arbetar tillsammans

13. Vilket eller vilka av följande påståenden beskriver UML bäst? (1p)

    1. Utan UML kan vi inte implementera ett program på ett objektorienterat vis, dvs. att vi måste minst skapa ett klassdiagram innan vi kan implementera någonting
    2. UML är den standard vars notation används för att modellera verkliga ting som första steget till ett objektorienterat program
    3. UML handlar om att skapa diagram av olika slag, dvs. att UML är samlingsnamnet för alla dessa diagram

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

16. Klassdiagrammet i *Figure 3* innehåller tomma klasser - er uppgift är att definiera relationer samt kardinalitet mellan dessa. Det finns flera sätt att lösa detta på så tänk på att dokumentera era val väl och förtydliga om så behövs (3p)

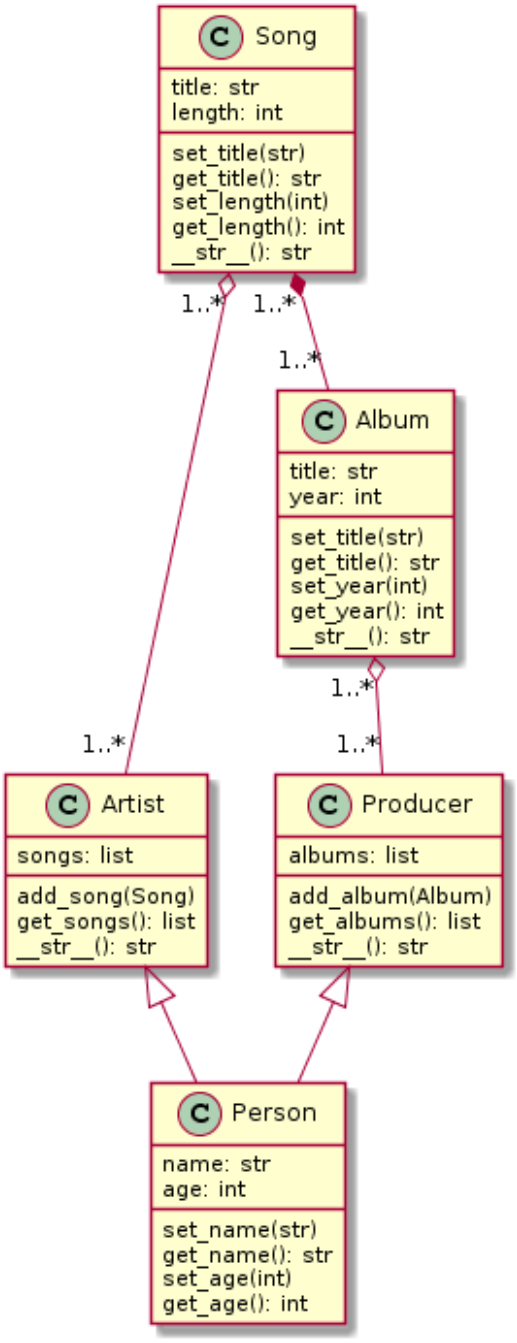

Figure 1: Klassdiagram för uppgift 14

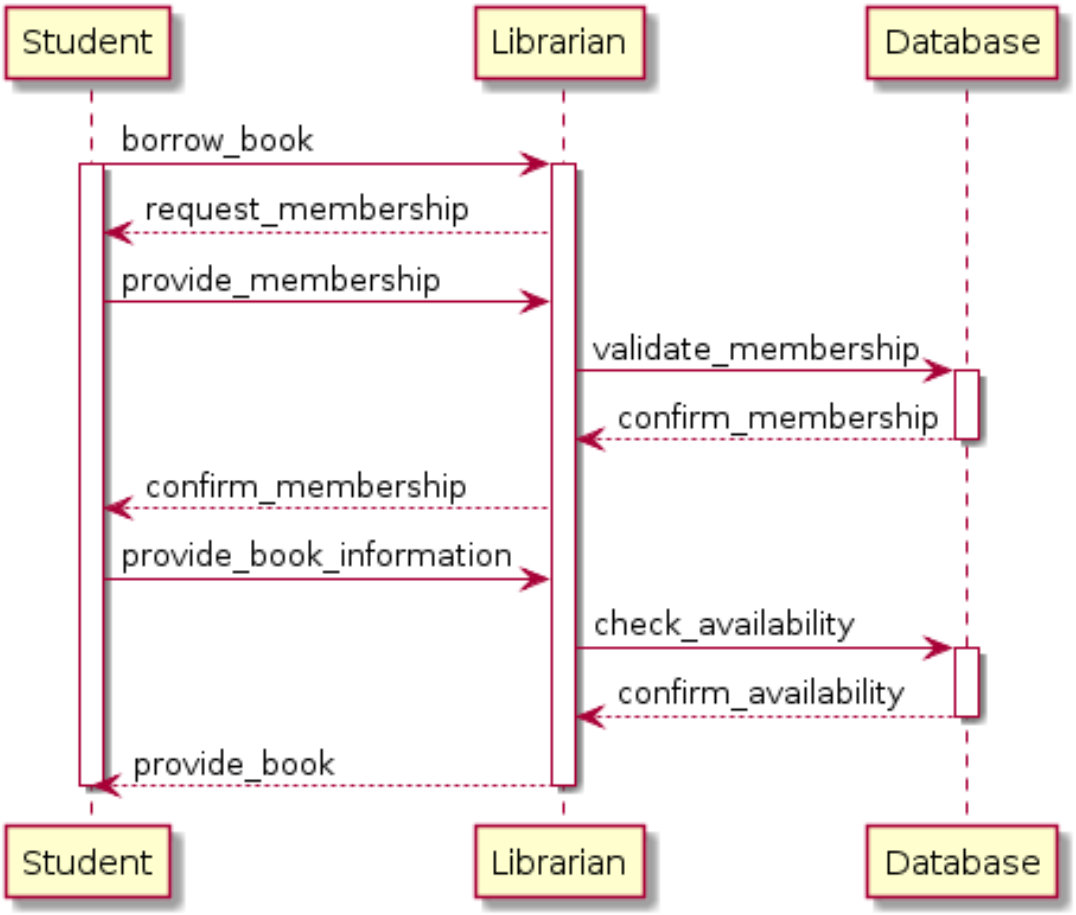

Figure 2: Sekvensdiagram för uppgift 15

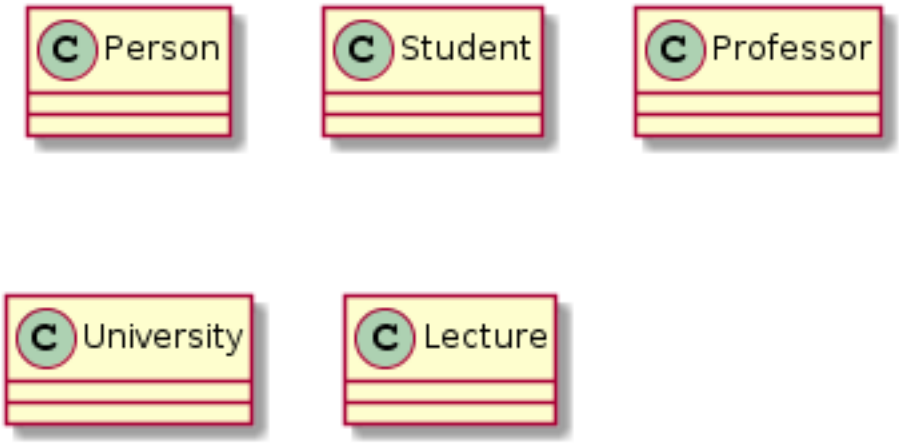

Figure 3: Klassdiagram för uppgift 16