Skolan för Datavetenskap och kommunikation

2D1311 Programmeringsteknik

Föreläsning 5

Kapitel 9

- Mer om klasser och objekt:
  - Speciella metoder
  - Meddelanden
  - Arv
  - Polymorfism
  - Rita objekt
  - Kortspelet Blackjack

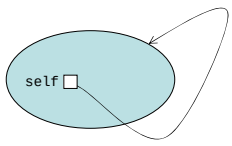
Klass och instans (objekt)

- En klass är en mall för ett objekt
  t ex bil, konto
- Ett objekt är en instans av en klass
  t ex den röda bilen, mitt lönekonto

Uppgift: Vilka av följande är klasser vilka är objekt?
a) En bok
b) Pythonboken
c) Min kopia av Pythonboken
d) Din hund
e) Bilden av din hund som biter brevbäraren
f) En bil
g) En Toyota Prius

self

Self är en referensvariabel som refererar från instansen till själva instansen.



Speciella metoder

- `__init__`
  Konstruktor - anropas när objekt skapas.
  Använd den för initiering av instansvariabler!
- `__str__`
  Låt den returnera en textrepresentation av objektet, så skrivs det ut med print.
- `__cmp__`
  Skriv en sån om du vill kunna jämföra två objekt med operatorerna > och <

Publika och privata metoder

- Om en metod definieras med ett namn som börjar med två understreck så är metoden *privat*. Det innebär att den endast kan anropas inom klassen (inte från kod utanför klassen).
- Om ett metodnamn inte definieras med två understreck så är metoden *publik*, och kan anropas från vilken del av programmet som helst.

Meddelanden

- När ett objekt anropar en metod hos ett annat objekt så säger man att första objektet *skickar ett meddelande* till det andra objektet (och det andra objektet tar emot ett meddelande från det första objektet).

```
class Students(object):

    def __init__(self,n):
        self.name = n

    def getName(self):
        return self.name

class Teacher(object):

    def __init__(self,n):
        self.name = n

    def introduce(self,s):
        favourites = s.getName()
        print "Mitt namn är",self.name,"och jag är stolt
                över mina studenter på", favourites

students = Students("Samhällsbyggnad")
teacher = Teacher("Linda Kann")
teacher.introduce(students)
```

Arv

- I objektorienterade språk finns möjligheten att använda arv.
- **Subklass**: Subklass är en klass som ärver från en annan klass.
- **Superklass**: Superklass kallas den klass som den andra klassen ärver från.
- En subklass ärver alla metoder och attribut från superklassen.

Överlagring och utökning

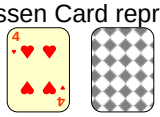
- *Överlagring* av en metod är då man definierar om en metod i en klass som klassen har ärvt från superklassen (Se geometri.py)
- Man *utökar* en subklass när man definierar nya metoder i subklassen som inte är definierade i superklassen.

Polymorfism

- Kommer av grekiskans πολλοι *(många)* och μορφη *(form)*
- Polymorfism är möjligheten att kunna skicka samma meddelande till objekt av olika klasser och få olika resultat.
- Metoden `__str__` som automatiskt anropas av print är ett exempel.

Spelkort

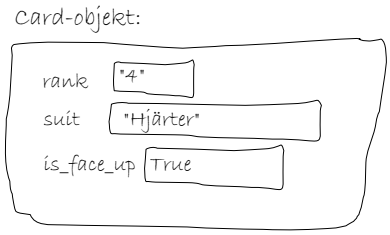
- Klassen Card representerar ett spelkort



- Klassens attribut är `rank` (valör), `suit`(färg) och `is_face_up` (uppvänt)
- Metoderna är `__init__` (konstruktorn), `__str__` (kortet som sträng), `flip` (vänder)

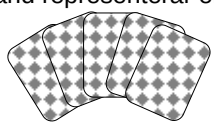
```
class Card(object):
    """ Ett spelkort. """
    RANKS = ["ess", "2", "3", "4", "5", "6", "7",
             "8", "9", "10", "knekt", "dam", "kung"]
    SUITS = ["Klöver", "Ruter", "Hjärter", "Spader"]

    def __init__(self, rank, suit, face_up = True):
        self.rank = rank
        self.suit = suit
        self.is_face_up = face_up

    def __str__(self):
        if self.is_face_up:
            rep = self.suit + " " + self.rank
        else:
            rep = "XX"
        return rep

    def flip(self):
        self.is_face_up = not self.is_face_up
```

Rita upp ett objekt



Korthand

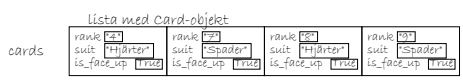
- Klassen Hand representerar en korthand



- Klassens attribut är `cards` (en lista med kort).
- Metoderna är `__init__`, `__str__`, `clear` (tömmer), `add` (lägger till ett kort), `give` (ger bort ett kort till en annan korthand)

```
class Hand(object):
    def __init__(self):
        self.cards = []

    def __str__(self):
        if self.cards:
            rep = ""
            for card in self.cards:
                rep += str(card) + "\t"
        else:
            rep = "<tom>"
        return rep

    def clear(self):
        self.cards = []

    def add(self, card):
        self.cards.append(card)

    def give(self, card, other_hand):
        self.cards.remove(card)
        other_hand.add(card)
```

Rita Hand-objekt



Kortlek

- *Arv* låter oss återanvända klasser!
- Vi vill ha en klass `Deck` som representerar en hel kortlek.
- Vi skriver `class Deck(Hand)`
- Klassen `Deck` ärver då alla attribut (`cards`) och metoder (`__str__`, `clear`, `add`, `give`) från Hand.
- Vi definierar också tre nya metoder i klassen `Deck`: `populate` (skapar ny lek), `shuffle` (blandar), `deal` (delar ut).

```
class Deck(Hand):
    """ En kortlek. """
    def populate(self):
        for suit in Card.SUITS:
            for rank in Card.RANKS:
                self.add(Card(rank, suit))

    def shuffle(self):
        import random
        random.shuffle(self.cards)

    def deal(self, hands, per_hand = 1):
        for rounds in range(per_hand):
            for hand in hands:
                if self.cards:
                    top_card = self.cards[0]
                    self.give(top_card, hand)
                else:
                    print "Slut på kort!"
```

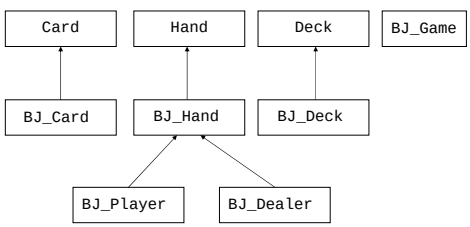
Spelregler för Blackjack

- I kortspelet Blackjack (även kallat tjugoett) är målet att summan av kortens valörer ska komma så nära 21 som möjligt.
- Kortsumman får inte överstiga 21 – då blir man tjock (och förlorar).
- Ess kan räknas som 1 eller 11.
- Knekt, Dam och Kung är alla värda 10.
- Alla spelare startar med två kort och kan välja att ta fler. Given måste ta fler upp till sjutton.

Klasser i Blackjack-programmet

Klass	Ärver från	Beskrivning
`BJ_Card`	`Card`	Som Card men med attributet `value` tillagt.
`BJ_Deck`	`Deck`	En lista med BJ_Card-objekt.
`BJ_Hand`	`Hand`	Som Hand men med attributen `total` och `name` tillagda.
`BJ_Player`	`BJ_Hand`	En blackjack-spelare.
`BJ_Dealer`	`BJ_Hand`	Given (som ger ut korten).
`BJ_Game`	`object`	Själva spelet! Har attributen `deck`, `dealer` och `players`.

Klassdiagram



Algoritm

- Ge två kort till varje spelare och given.
- Låt varje spelare:
  - få ett kort till, så länge den vill ha ett kort till och inte blivit tjock
- Om alla spelare blivit tjocka:
  - visa givens två kort
- Annars:
  - låt given få ett kort till, så länge den har mindre än sjutton (och inte blivit tjock)
- Om given blivit tjock:
  - så vinner alla kvarvarande spelare
- Annars:
  - Gå igenom varje kvarvarande spelare:
    - spelare som ligger högre än given vinner, spelare som ligger lägre förlorar