Universidade Federal de Minas Gerais
Ciência da Computação

Linguagens de Programação - Haniel Barbosa

Lista de Exercícios 4

Elaborada por José Wesley Magalhães

1. Nesta questão você deve implementar um Tipo Abstrato de Dado em SML para representar uma biblioteca Math que lida com números inteiros e fornece algumas funções para o usuário. Você deve definir tanto a especificação quanto a implementação. O nome de sua *structure* deve ser `MyMathLib`, e você deve implementar quatro operações:

   - fact: calcula o fatorial de um número
   - halfPi: constante representando metade do valor de Pi
   - pow: dado uma base e um expoente, calcule a potência
   - double: dobra um número

   NOTA: o valor `halfPi` deve ser real.

   input: MyMathLib.pow(2,3)
   output: val it = 8 : int

   input: MyMathLib.double(6)
   output: val it = 12 : int

2. Utilizando a classe `Node` descrita abaixo, defina, em Python, um tipo abstrato de dado `Stack` que armazena objetos do tipo `Node`. O campo `e` armazena uma `string` e o campo `n` aponta para o próximo elemento na pilha.

```
1 class Node:
2   def __init__(self):
3     self.n = 0
4     self.e = ''
```

   Você deve implementar os seguites métodos:

   - add: adiciona um `Node` na pilha
   - remove: remove um `Node` da pilha e retorna o elemento desse `Node`
   - isNotEmpty: retorna `True` se a pilha não é vazia, e `False` caso contrário

   O construtor deve iniciar o topo da pilha com um `Node` vazio.

   Exemplo:

```
>>> s = Stack()
>>> s.add("Baltimore")
>>> s.add("Lord")
>>> s.add("Sir")
>>> s.isNotEmpty()
True
>>> while (s.isNotEmpty()):
     print (s.remove())
Sir
Lord
Baltimore
```

3. Utilizando a mesma classe `Node`, defina agora, também um Python, um tipo abstrato de dado `Queue` que implemente os mesmos métodos que `Stack` mais um método `getSmaller()`, o qual retorna o menor elemento da fila. Note que este método apenas retorna esse elemento, não o remove da fila. Você pode modificar a classe `Node` para esta implementação desde que ela ainda funcione para o TAD `Stack`. **NOTA:** para comparar strings lexicograficamente, utilize os operadores relacionais: $<, >, \leq, \geq, =, \neq$.

   Exemplo:

```
>>> q = Queue()
>>> s.add("C")
>>> s.add("A")
>>> s.add("B")
>>> s.isNotEmpty()
True
>>> s.getSmaller()
'A'
```

4. Considere o método `removeAll` abaixo e responda:

```
1 def removeAll(s):
2   """Removes all the elements from the data structure."""
3   while (s.isNotEmpty()):
4     print (s.remove())
```

   (a) Qual o "contrato" que deve ser garantido pelos objetos passados para este método? Isto é, pelos elementos de `s` passado para o método?

   (b) O que significa a expressão *duck typing*? E qual sua relação com este método?

5. Considere o programa abaixo e responda o que acontecerá em cada linha numerada. As opções possíveis são:

   (i) Algo será impresso. Neste caso, escreva o que será impresso.

   (ii) Um erro será produzido em tempo de execução.

```
1  class Animal:
2      def __init__(self, name):
3          self.name = name
4      def __str__(self):
5          return self.name + " is an animal"
6
7      def eat(self):
8          print (self.name + ", which is an animal, is eating.")
9
10 class Mammal(Animal):
11     def __str__(self):
12         return self.name + " is a mammal"
13
14     def suckMilk(self):
15         print (self.name + ", which is a mammal, is sucking
               milk.")
16
17 class Dog(Mammal):
18     def __str__(self):
19         return self.name + " is a dog"
20
21     def bark(self):
22         print (self.name + " is barking rather loudly.")
23
24     def eat(self):
25         print (self.name + " barks when it eats.")
26         self.bark
27
28 def test():
29     a1 = Animal("Pavao")
30     a2 = Mammal("Tigre")
31     a3 = Dog("Krypto")
32     print (a1)              # 1
33     print (a2)              # 2
34     print (a3)              # 3
35     a1.eat()              # 4
36     a2.suckMilk()         # 5
37     a2.eat()              # 6
38     a3.bark()             # 7
39     a3.suckMilk()         # 8
40     a3.eat()              # 9
41     a1.bark()             # 10
42     a1 = a3
43     a1.bark()             # 11
```

6. Acerca de orientação a objetos, descreva o que é o "Problema do Diamante".

7. Nesta questão, você deve adicionar exceções no TAD criado na questão 1. Você deverá restringir todos os valores manipulados pelas funções definidas a números positivos. Você também deverá criar uma função `useMyMathLib : int * string -> unit` que utiliza os métodos de `MyMathLib` e imprime o resultado das operações utilizando a função `print` de SML. O primeiro parâmetro é um valor a ser usado nas funções e o segundo é uma string com a função a ser usada. No caso de `pow`, suponha que sempre estaremos elevando um valor x a x^x. Essa função deve tratar as exceções disparadas por `MyMathLib`, exibindo a mensagem "Não posso lidar com valores negativos!". Você é livre pra modificar a implementação da questão 1 como achar melhor, desde que não modifique o comportamento esperado.

   input: useMyMathLib(2, "pow")
   output: 4 val it = () : unit

   input: useMyMathLib(∼3, "fact")
   output: Não posso lidar com números negativos val it = () : unit

8. Nesta questão você deverá escrever uma calculadora interativa em Python, a qual recebe pela entrada padrão operações com $+, -, *, /$ em formato de string. Você deverá tratar os seguites erros:

   - Se a entrada não consistir de 3 elementos, dispare uma `FormulaError`, que é uma exceção customizada com a mensagem "A entrada nao consiste de 3 elementos".
   - Tente converter a primeira e a segunda entrada para `float`, trate cada `ValueError` que acontecer e dispare uma `FormulaError` com a mensagem "O primeiro e o terceiro valor de entrada devem ser numeros".
   - Se o segundo elemento não for nenhum dos operadores aritméticos descritos acima, dispare uma exceção com a mensagem "x nao e um operador valido".

   Exemplo:

```
>>> 1 + 1
2.0
>>> 1 +
Traceback (most recent call last):
  File "8.py", line 35, in <module>
    n1, op, n2 = parse_input(user_input)
  File "8.py", line 8, in parse_input
    raise FormulaError('A entrada nao consiste de 3 elementos')
__main__.FormulaError: A entrada nao consiste de 3 elementos
>>>
```