Moving to J2SE 5.0

Tiger Hunting with Java 5

Exercise book

Version 1.0

02-03 Boxing & Varargs

Testing Infrastructure

Write a simple test execution framework that can execute many tests using Varargs. Write a Test interface and a TestSuite class. TestSuite should have a execute() method that gets one or more test for execution.

Here is an example of a testing code based on the platform:

```
class TestAAA() implements Test { ... }
class TestBBB() implements Test { ... }

Test ta = new TestAAA();
Test tb = new TestBBB();

TestSuite st = new TestSuite();
st.execute(ta,tb,tb);
```

Boxing Performance Evaluation

With boxing it is possible to use wrapper classes for performing calculations.

For example:

```
Integer oi = new Integer(0);
oi++;
```

Doing so introduce a significant performance overhead compared to the alternative of doing the same thing using primitive types:

```
int pi = 0;
pi++;
```

(a) Write test that measures the time it takes to performing 1,000,000 increase operations using
  (1) class Integer
  (2) int primitive

(b) Compare the execution time for the two alternatives

(c) Use your code to compare performance of more operations on Integer vs. int such as:

```
oi.equals()      vs    pi==
""+oi            vs    ""+pi
oi.toString()    vs    Integer.toString(pi)
oi+=1            vs    pi+=1
```

04 For/In

Inherit the StringTokenizer class and make it Iterable (by implementing the Iterable interface). Test your new class within a for/in statement.

05 Generics

(A) Write a generic Cache, the Cache is used to hold objects of a specific type according to a key of another specific type. The size of the cache is pre-defined.

Users should be able to:
(1) offer a key value to the cache
(2) get objects from the cache according to a key (should return null if the object does not exists)
(3) invalidate a cache entry according to a key.

Here is a usage example:

```
Cache<Integer,String> c = new Cache<Integer,String>(2);

c.offer(1,"*One*");
c.offer(2,"*Two*");
c.offer(3,"*Three*");

for (int i=0;i<4;i++)
     System.out.format("%d:%s\n",i,c.get(i));
```

(B) Write a generic Pool, the Pool creates N instances of a specific type.

Users should be able to:
(1) get an object from the pool (or null if the pool is empty)
(2) return objects to the pool

Important: The only way for the Pool to be able to construct objects of the generic type safely is to pass the type Class object to the Pool's constructor.

Here is a usage example:

```
Pool<BigClass> p = new Pool<BigClass>(BigClass.class,3);
BigClass o = p.get();
o.doSomething();
p.free(o);
```

06 Formatting

Use format() and GregorianCalendar() to write a program that outputs an ASCII calendar.

Output should look like this:

```
January 2004
Sun Mon Tue Wed Thr Fri Sat
                  1   2   3
  4   5   6   7   8   9  10
 11  12  13  14  15  16  17
 18  19  20  21  22  23  24
 25  26  27  28  29  30  31
```

07 Enums

Write an implementation for a type-safe deck of cards.

A card should have a rank (one of: TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING or ACE } and a suit (one of DIAMOND, HEART, SPADE or CLUB)

A deck is simply a List of Cards.

08 Threading

Write a simulation for the card game "speed", the rules are:

- A deck of cards is divided among four players

- Each player drops one card into an empty pile
- Each player can put a card on any pile if the card number is the one before or the one after the card on the top of the pile
- The first player that drops all his cards wins!

Each Player should be executed using its own thread (for parallelism). For the purpose of the exercise it is ok if all the players will use the same play strategy, for example:

```
For each Pile
        For each Card in myDeck
                Put card on pile if possible.
```

You must implement some thread synchronizing to prevent two players from putting two cards on the same pile at the same time.