CS 367 Homework 1

Mingren Shen (mshen32@wwisc.edu)

Question 1

Consider the following code fragment that is intended to remove every other item from a `List<String>` object. For example, given the list `[ "Ben", "Beck", "Eric", "Vidya", "Jason", "Xiaoming" ]`, it should remove `"Ben"`, `"Eric"`, and `"Jason"` and leave the list as just `[ "Beck", "Vidya", "Xiaoming" ]`.

```
for (int i = 1; i <= words.size(); i += 2) {
    words.remove(i);
}
```

A. If words initially contains ["washington", "j. adams", "jefferson", "madison", "monroe", "j.q. adams", "jackson", "van buren"]` then what is left in words after this code is done running?

After the running, the words is now `[ "washington", "jefferson", "madison", "j.q. adams", "jackson" ]`

B. Does this code work correctly? If not, briefly describe (in a sentence or two) how it differs from what we said it should do.

No, it does not work correctly.

If you remove an entry from the list while you are still looping over it, removing the element reduces the length of the list by one already. So you only need to increase your index by 1 only.

Then the code has some detailes that needs corrections.

(1) `i` should start from 0 instead of 1.

(2) the condition should be `i` is smaller than the size of the list not equal to, other than there will be out of index exceptions.

C. If the code is broken, describe how to fix it.

So the right code should be

```
for (int i = 0; i < words.size(); i += 1)
{
    words.remove(i);
}
```

Question 2

Assume that the `ArrayList` constructor initially reserves enough room to store ten elements without resizing. When resizing is needed, assume that the larger array is three times as big as the old one.

(1) Suppose we have called `ArrayList.add` 200 times on a single list, and have never removed any of the added elements. How many times has the array been resized? How many more elements can we add (with no removals) *without* causing the array to resize again?

We should resize the array 3 times.

And the capacity now is $3^3 \times 10 = 270$, then the remeaning space is 270 - 200 = 70.

(2)More generally, if an `ArrayList` has an initial capacity of *i* elements, and the array is tripled in size *d* times, then what will the total capacity be after these triplings?

The total capacity is $i \times 3^d$

(3)Conversely, if an `ArrayList` has an initial capacity of *i* elements, then how many triplings are needed to reach a total capacity of at least *c* elements?

The number of tripling is $log_3(c/i)$ and the result should be ceil to the nearest whole number. So the number of tripling is the ***least*** integer that is greater than or equal to $log_3(c/i)$.

Question 3

Consider the following pseudo-code:

```
void main( ) {
    println("main enter");
    a();
    try {
        c();
```

```
        if (v4 == true) throw new Exc4();
    } catch (Exc4 ex) {
        println("main caught Exc4");
    } catch (Exc2 ex) {
        println("main caught Exc2");
    } catch (Exc1 ex) {
        println("main caught Exc1");
    }
    println("main exit");
}

void a( ) {
    println("a enter");
    try {
        b();
    } catch (Exc1 ex) {
        println("a caught Exc1");
    }
    println("a exit");
}

void b( ) {
    println("b enter");
    if (v2 == true) throw new Exc2();
    println("b exit");
}

void c( ) {
    println("c enter");
    try {
        d();
        if (v4 == true) throw new Exc3();
    } catch (Exc3 ex) {
        println("c caught Exc3");
        if (v3 == true) throw new Exc4();
    } catch (Exc4 ex) {
        println("c caught Exc4");
    }
    println("c exit");
}

void d( ) {
    println("d enter");
    if (v1 == true) throw new Exc1();
    if (v3 == true) throw new Exc3();
    if (v5 == true) throw new Exc5();
```

```
    println("d exit");
}
```

For the each part below **determine the complete output** that would be generated if the pseudo-code above was run with the values of the `v` variables as specified below. Assume the exception classes `Exc1`, `Exc2`, `Exc3`, `Exc4`, and `Exc5` each extend `RuntimeException`. If an exception is passed out of main, show the output of the runtime environment as "`Program terminated due to Exception Exc*N*` ", where *N* is the particular exception number.

A. What would be output if `v1` is true and the other variables are false?

The output is:

```
main enter
a enter
b enter
b exit
a exit
c enter
d enter
main caught Exc1
main exit
```

B. What would be output if `v2` is true and the other variables are false?

The output is:

```
main enter
a enter
b enter
Program terminated due to Exception Exc2
```

C. What would be output if `v3` is true and the other variables are false?

The output is:

```
main enter
a enter
b enter
b exit
a exit
c enter
d enter
c caught Exc3
main caught Exc4
main exit
```

D. What would be output if `v4` is true and the other variables are false?

The output is:

```
main enter
a enter
b enter
b exit
a exit
c enter
d enter
d exit
c caught Exc3
c exit
main caught Exc4
main exit
```

E. What would be output if `v1` and `v4` are true and the other variables are false?

The output is:

```
main enter
a enter
b enter
b exit
a exit
c enter
d enter
main caught Exc1
main exit
```

F. How would the code need to be modified if exception type `Exc2` were a *checked* exception?

So a checked exception must be decleard in the header of a method. For this question, we will need to add `throws Exc2` to the header of `main()`, `a()`, `b()` methods.

And for the main block we will need to remove the unreachable catch block for Exc2. This exception will never be thrown from the try statement body.

G. How would the code need to be modified if exception type `Exc5` were a *checked* exception?

The same as F above, we only need to add `throws Exc5` to the header of method `main()`, `c()`, `d()`.