Moebius Solutions Application Questions

**================**

> VERSION: 2021-01-02

Name of Candidate: Ramon Ortega

QUESTION 1. Why does this code not work? How do you fix it?

**----------------**

```

public class Example {

    public List<Integer> removeBigNumbers(List<Integer> data) {

        for (Integer i : data) {

            if (i > 10) {

                data.remove(i);

            }

        }

        return data;

    }

}

```

QUESTION 2: Programming Task

**----------------**

Write a class that implements the following interface, assuming that all methods

are used with approximately the same frequency.

```

interface ItemStore {

    Interface Item {

        String getID();

        String getTag();

    }

    /\*\*

     \* Adds an {@link Item} to the store, replacing any existing item with the

     \* same {@link Item#id} value.

     \*/

    public void put(Item item);

    /\*\*

     \* Retrieves the {@link Item} with the given {@link Item#id} value, or

     \* null if no such {@link Item} exists in the store.

     \*/

    public Item get(String id);

    /\*\*

     \* Delete all {@link Item}s with the specified tag.

     \*/

    public void dropAllByTag(String tag);

    /\*\*

     \* Returns the number of Items in the store

     \*/

     public int size();

}

```

QUESTION 3: Memory Management

**----------------**

The `SmallMemoryMessageTest` class below passes on our development machines,

but the client reports that it fails on their 64MB VM.

1. Run on a 64M VM and copy the failing stack trace.

2. Modify `main()` to work on a 64MB VM.

3. Ensure there are no more than one performance warnings.

To set 64Mb VM, run using `java -Xmx64M SmallMemoryMessageTest`.

```

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import java.util.function.Predicate;

/\*\*

\* The following mock program passes on our development machines,

\* but the client reports that it fails on their 64MB VM.

\*

\* 1. Run on a 64M VM and copy the failing stack trace

\* 2. Modify main() to work on a 64MB VM.

\* 3. Ensure at most one performance warning.

\*

\* (To set 64Mb VM) run using java -Xmx64M SmallMemoryMessageTest

\*/

/\*\* A given interface to process messages (DO NOT CHANGE) \*\*/

interface MessageProcessor {

void processMessage(Message msg);

}

/\*\* A given interface to archive select messages (DO NOT CHANGE) \*\*/

interface MessageArchiver {

void archiveMessages(List<Message> messages, Predicate<Message> filter);

}

/\*\* A given class to represent a message (DO NOT CHANGE) \*\*/

class Message {

private String subject;

private String body;

public Message(String subject, String body) {

this.subject = subject;

this.body = body;

}

public String getSubject() {

return subject;

}

public String getBody() {

return body;

}

@Override

public String toString() {

return "Message{" +

"subject: " + Util.abbreviate(subject, 20) +

", body: " + Util.abbreviate(body, 40) + "}";

}

}

/\*\*

\* Test class that works on developer boxes but fails on small VMs.

\*

\* Fix this class to work on 64Mb VM

\*/

public class SmallMemoryMessageTest {

public static void main(String []args) {

MessageProcessor processor = Util.createMessageProcessor();

MessageArchiver archiver = Util.createMessageArchiver();

List<Message> messages = new ArrayList<>();

for (int i = 0; i < Util.EXPECTED\_TOTAL; i++) {

Message msg = Util.random();

processor.processMessage(msg);

messages.add(msg);

}

archiver.archiveMessages(messages, m -> m.getSubject().startsWith("A"));

/\*

\* DO NOT CHANGE ANYTHING BELOW THIS LINE.

\* PROGRAM MUST EXIT SUCCESSFULLY

\*/

Util.validate();

}

}

/\*\* A given utility class (DO NOT CHANGE) \*\*/

class Util {

static final int EXPECTED\_TOTAL = 98765;

static final int EXPECTED\_ARCHIVED = 3799;

static Message random() {

String subject = randomSubject();

String body = randomBody();

Message m = new Message(subject, body);

return m;

}

static int count = 0;

static String randomSubject() {

StringBuilder sb = new StringBuilder(128);

sb.append((char) ((int) 'A' + (count++ % 26)));

Random r = new Random();

while (sb.length() < 128) {

char c = (char) r.nextInt(127);

if (Character.isLetterOrDigit(c)) {

sb.append(c);

}

}

return sb.toString();

}

static String randomBody() {

StringBuilder sb = new StringBuilder(4096);

Random r = new Random();

while (sb.length() < 4096) {

char c = (char) r.nextInt(127);

if (!Character.isISOControl(c)) {

sb.append(c);

}

}

return sb.toString();

}

static String abbreviate(String s, int n) {

return s.length() > n + 3 ? s.substring(0, n - 3) + "..." : s;

}

static MessageProcessor createMessageProcessor() {

return new TestProcessor();

}

static MessageArchiver createMessageArchiver() {

return new TestArchiver();

}

static void validate() {

if (TestArchiver.count != EXPECTED\_TOTAL ||

TestArchiver.archived != EXPECTED\_ARCHIVED) {

throw new IllegalStateException("Failed to archive all messages!");

}

System.out.println("SUCCESS. PROCESSED: " + TestArchiver.count + " ARCHIVED: " + TestArchiver.archived);

}

}

/\*\* Noop implementation of a message processor (DO NOT CHANGE) \*\*/

class TestProcessor implements MessageProcessor {

long count = 0;

@Override

public void processMessage(Message msg) {

if(++count%1000 == 0) {

System.out.println("Processed: " + (count) + " Latest: " + msg);

System.out.flush();

}

}

}

/\*\* Noop implementation of a message archiver (DO NOT CHANGE) \*\*/

class TestArchiver implements MessageArchiver {

static int count = 0;

long bytes = 0;

static long archived = 0;

@Override

public void archiveMessages(List<Message> messages, Predicate<Message> filter) {

if(messages.size() < 1000) {

System.err.println("WARNING: message list too short, this will drastically reduce performance!");

}

messages.stream().filter(filter).forEach(this::archiveOne);

System.out.println("Archived: " + bytes + " bytes.");

count += messages.size();

bytes = 0;

}

private void archiveOne(Message msg) {

bytes += msg.getSubject().getBytes().length + 1;

bytes += msg.getBody().getBytes().length;

archived++;

}

}

```

QUESTION 4: Debugging

**----------------**

```

package com.moesol.hr.bugs;

public class Bug1 {

    private Integer rating;

    public int rating() {

        return rating;

    }

    public static void main(String[] args) {

        System.out.println("rating:"

            + new Bug1().rating());

    }

}

```

The program above throws a `NullPointerException` with this stack trace:

```

Exception in thread "main" java.lang.NullPointerException

    at com.moesol.hr.bugs.Bug1.rating(Bug1.java:7)

    at com.moesol.hr.bugs.Bug1.main(Bug1.java:12)

```

What is happening? How can it be fixed?

QUESTION 5: Wrong Result

**----------------**

The following program produces inconsistent results. It should always output

this:

```

counter is 20000

```

Please correct the program.

```

public class WrongAnswer {

    private int counter = 0;

    public static void main(String[] args) {

        new WrongAnswer().run();

    }

    private void run() {

        try {

            Thread t1 = new Thread(this::incrementToOnHundred);

            Thread t2 = new Thread(this::incrementToOnHundred);

            t1.start();

            t2.start();

            t1.join();

            t2.join();

            System.out.println("counter is " + counter);

        } catch (InterruptedException e) {

            System.err.println("fatal error, unexpected interrupt exception");

            System.exit(2);

        }

    }

    private void incrementToOnHundred() {

        for (int i = 0; i < 10\_000; i++) {

            doSomeFakeWork();

        }

    }

    private void doSomeFakeWork() {

        counter++;

    }

}

```

BONUS: Programming Puzzle Bonus!

**----------------**

A string of ASCII text has been encrypted using the following algorithm:

\* Use the bytes "MOEBIUS" as the key.

\* Working one byte at a time, xor the next byte of data with the next byte in

the key.

\* When the end of the key is reached, wrap around to the first byte again.

\* Base-64 encode the data.

For example, the text "Simple is better than complex." encrypts to

"HiYoMiUwcyQ8ZSAsIScoPWU2ITQ9bSwqLzk5NjVh". The key is repeated as follows

while XOR-ing bytes:

```

Simple is better than complex.

MOEBIUSMOEBIUSMOEBIUSMOEBIUSMO

```

Write a program to reverse this process and decrypt the following message. Use

any programming language you like. Provide both your program and the decrypted

message.

```

DiArJTs0JzgjJDYgOj0+Y2U7JiBzPiApNCwxczknIGIrOj04PGUzPDAgOSYqLGhfWW1vZW

JpdXNtb2ViaXVzbW9lYml1c21vZWJpdXNtb2ViaXVzY2hibEN1c21vZWJpe3RqYWViaXVz

bWFlYml1c21vZWhucnltb2VicwoPYhB/Yml1c21hT2JpdXNtb38dFXoMd29lYhYJe2IQZW

Jnb31nEBltFn9zbW9/YmYJc3dvZWxue2ljaGtIaXV9amhreGl6D211ZWJpe3xkE2ViaXJp

amVlbRV1eW11ZWJue31qYWViZGhpInV4b0N1aRITah1zcn13dX9saXVzbWhlaG5yeW1vZW

JjdXRjaGpsbnUMEWdqHW57dHdoa2VDdWltYBlic3Vpd3V/eGl1c21vbx0Vegxnb2ViaXV+

cG8qYnR4c21gbB5pdXNtaE9iaXJ9Y2hlYm5vaXdoZWJpdXNnb2oeaX9zbW9lYmdyfGNoa2

JpdXRHb2ViaXVzbW9lYml1c21vZWJpdXljYW9iaXVzbW9lYmlvWQ==

```

> NOTE: This block uses the "MIME" flavor of Base64 encoding.

> Depending upon your Base64 decoder, you may need to remove the "=" padding.

>