Technical Assessment – Target.com

Programming Case Study – Browse/ Front facing app

myRetail is a rapidly growing company with HQ in Richmond, VA and over 200 stores across the east coast. The stores average 80,000 sq. ft. in size and carry around 20,000 products. myRetail wants to provide a multi-channel experience for its customers online.

myRetail is comparing solution options for the online grocery store. The goal for this exercise is to create an end-to-end POC for a Products service and API, which will return Product data from the database to the caller. You are free to use frameworks, app servers and storage solutions of your choice. For example, Apache/Tomcat and MySQL would be possible options. Here is some example data:

id	sku	name	category	last_updated
5555	AEX143	Stroller	baby	2014-05-23
5543	IOL123	Optimus Prime	toys	2014-02-01
7563	XYZ904	Sega Genesis	toys	1989-04-01

id	price
5555	199.99
5543	13.37
7563	149.99

Your goal is to create a RESTful service that, for can retrieve product and price details by ID. The URL structure is up to you to define, but try to follow some sort of logical convention.

Create a Java project that does the following:

1) **Use an ORM to abstract the DB layer from the server side code, and read data from the database. You can use myBatis or another framework of your choosing.**
2) **Write JUnit test cases to verify you can read and write from the database tables.**
3) **Create a service class that will return product details by ID as described above. You are free to use a framework (like CXF, Axis2, etc), or just plain servlets if you wish.**
4) **Write JUnit test cases to verify your service functionality.**
5) **BONUS: Add the capability to retrieve multiple products at once, or retrieve a list of details by category, and write associated JUnit test cases.**
6) **Write a short paragraph describing the structure of your code, relevant classes, etc.**

<u>Deliverables</u>

- **Send in the project archive, including the source code and all the associated libs and server config**

Programming Case Study – General Analytical Programming

You have a farm of 400m by 600m where coordinates of the field are from (0, 0) to (399, 599). A portion of the farm is barren, and all the barren land is in the form of rectangles. Due to these rectangles of barren land, the remaining area of fertile land is in no particular shape. An area of fertile land is defined as the largest area of land that is not covered by any of the rectangles of barren land.

Input

You are given a set of rectangles that contain the barren land. These rectangles are defined in a string, which consists of four integers separated by single spaces, with no additional spaces in the string. The first two integers are the coordinates of the bottom left corner in the given rectangle, and the last two integers are the coordinates of the top right corner.

Output

Output all the fertile land area in square meters, sorted from smallest area to greatest, separated by a space.

Sample Input/ Output

Sample Input	Sample Output
{“0 292 399 307”}	116800 116800
{“48 192 351 207”, “48 392 351 407”, “120 52 135 547”, “260 52 275 547”}	22816 192608

Deliverables

- Code

```
import java.io.*;
public class Solution {
    public static void main(String args[] ) throws Exception {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    }
}
```

-

Programming Case Study – Search

The goal of this exercise is to create a working program in Java to search relevant documents / files for the given search term or phrase (single token), and return results in order of relevance. Relevancy is defined as number of times the exact term or phrase appears in the document.

We have attached 3 files for your program to read and use as sample search content. Your project should allow the user to enter a search term or phrase at the command-line, execute the search, and return results. For example:

```
Enter the search term: <user enters search term>
Search results:
        <sample results start:
        File2.txt – X matches
        File1.txt  - X matches
        File3.txt – X matches
        >
Enter the search term: <user enters search term>
```

For this project, create a modular "searcher" class to execute the search against the content. You will have to create 3 iterations of this "searcher," each with a different approach:

- Brute-force text searcher
- Text search using the regex API in Java
- Create an "index" of content using a data structure, like a hash map or binary tree.
- Write unit tests for each method
- Run a performance test that does 2M searches with random search terms, and measures execution time. Which approach is fastest? Why?
- Provide some thoughts on what you would do on the software or hardware side to make this program scale to handle massive content and/or very large request volume (5000 requests/second or more).

Deliverables

1. **Working program and tests**
2. **Performance results and analysis**
3. **Thoughts on scalability**

Deliverables

- Rename .js files to .js.txt before zipping up all code and attach to an email to **jessi.wallace@target.com** two days prior to the your onsite interview
- Bring a print out of the code to the interview.