Java Take-Home Interview Questions

Only do one, not both.

Question 1 (for beginning/intermediate developers):

Write a Java program to process lines of input.

```
package com.hobsons.interview;
public class ProcessLines {
 public void process(List<String> lines) {
  //code goes here
 }
}
```

Each line has two parts, Category and Subcategory.

The valid categories are:
FOOD
PERSON
PLACE

Print out the following:
1. The input lines, with invalid categories and duplicates removed
2. A chart that lists the categories and the number of unique lines with those categories

Example:

Input lines:

```
PERSON Bob Jones
PERSON Fred Smith
FOOD Pizza
PET Fido
FOOD Ice Cream
PLACE New York
FOOD Pizza
TREE Maple
PERSON George Washington
PLACE Omaha
```

Output:

```
PERSON Bob Jones
PERSON Fred Smith
FOOD Pizza
FOOD Ice Cream
PLACE New York
PERSON George Washington
PLACE Omaha
FOOD 2
PERSON 3
PLACE 2
```

Your code must include automated tests to validate the functionality implemented.

You must provide a build file that downloads dependencies, such as Ant+Ivy, Maven, or Gradle; do not package third-party dependencies with your submission.

Submit your application by zipping up your build file, source code, and tests into a single archive. Do not post your application to Github or to any other public source code repository.

Question 2 (for advanced/architect/lead/etc.)

Build a Java application to interact with the web service described below.

Implementation information

Your application must include a class that implements the following interface:

```
package com.hobsons.interview;
public interface PeopleProcessor {
 /**
 This method takes in a filename, opens the CSV file specified, processes
 the contents, and returns a list of the Person objects currently in the
 system.
 */
 List<Person> processFile(String fileName);
}
```

The Person class looks like this:

```
package com.hobsons.interview;
public class Person {
 private int id;
 private String name;
 private int age;


 public Person() {
 }


 public void setId(int id) {
  this.id = id;
 }


 public int getId() {
  return id;
 }

 public void setName(String name) {
  this.name = name;
 }

 public String getName() {
  return name;
 }

 public void setAge(int age) {
  this.age = age;
 }

 public int getAge() {
  return age;
 }
}
```

The CSV has the following format:

```
CODE,data[,data2,data3]
```

The valid values for CODE are:

A - Add a Person
U - Update a Person
D - Delete a Person

The data for each code is:
A - name,age
U - id,name,age
D - id

For example:

```
A,Bob Jones,10
A,Fred Smith,20
U,5,George Jackson,50
D,6
```

If an entry in the file does not match this structure, log it.

If an entry refers to a non-existent ID, log it.

Your code must include automated tests to validate the functionality implemented.

Please note that the list of people in the system is not permanent, and could be wiped out at any time.

You must provide a build file that downloads dependencies, such as Ant+Ivy, Maven, or Gradle; do not package third-party dependencies with your submission.

Submit your application by zipping up your build file, source code, and tests into a single archive. Do not post your application to Github or to any other public source code repository.

API

note: any place you see :id, replace it with the actual ID for a person. For example, to get the user with ID 5, you would send a GET to:

http://interview.starfishsolutions.com:8000/person/5

Get all people

GET http://interview.starfishsolutions.com:8000/person

response

status code: 200 - OK
content:

```
{"list": [{"id": integer, "name": "Person's Name", "age": integer}]}
```

example

```
{
 "list": [
  {
   "id": 1,
   "name": "Joe Smith",
   "age": 20
  },
  {
   "id": 2,
   "name": "Bob Jones",
   "age": 10
  },
  {
   "id": 3,
   "name": "Wanda West",
   "age": 50
  }
 ]
}
```

Get a single person

GET http://interview.starfishsolutions.com:8000/person/:id

response

status code: 200 - OK
content:

```
{"id": integer, "name": "Person's Name", "age": integer}
```

example

```
{
 "id": 2,
 "name": "Bob Jones",
 "age": 10
}
```

Create a single person

POST http://interview.starfishsolutions.com:8000/person

body

```
{"name": "Person's Name", "age", integer}
```

example

```
{
 "name": "Bob Jones",
 "age": 10
}
```

note: if the id is included in the body, it will be ignored.

response

201 - Created
Location Header with URL for new resource

Update a single person

PUT http://interview.starfishsolutions.com:8000/person/:id

body

`{"name": "Person's Name", "age", integer}`

example

```
{
 "name": "Bob Jones",
 "age": 15
}
```

note: if the id is included in the body, it will be ignored.

response

200 - OK

Delete a single person

DELETE http://interview.starfishsolutions.com:8000/person/:id

response

204 - No Content

Error codes

404 - id doesn't exist for a GET, PUT, DELETE
400 - bad request sent by user (invalid person on POST/PUT, non-integer ID on GET/PUT/DELETE)
500 - error building JSON on server (shouldn't happen)

When an error code is returned for the status, the body of the response will contain
the error JSON:

```
{"error":"Error Message"}
```