DevSkiller TalentScore

Evaluation details 7D4T-6AFP-QHW6

Created on: 2021-09-28

grzegorzmalarski@wp.pl

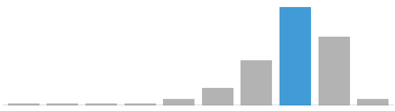
Results overview

Test Score

70 %

35 of 50 points

Score comparison



Worse than 30.97% of other candidates.

Skills

JAVA - 58% (18 / 31)

SPRING - 78% (7 / 9)

SQL - 86% (6 / 7)

HIBERNATE - 67% (4 / 6)

GIT - 50% (2 / 4)

QA - 100% (4 / 4)

JAVA 8 - 75% (3 / 4)

CLEANCODE - 100% (3 / 3)

API - 0% (0 / 2)

SPRING - DEPENDENCY INJECTION - 100% (2 / 2)

UNIT-TESTING - 100% (2 / 2)

SPOCK - 100% (1 / 1)

JUNIT - 100% (1 / 1)

MYSQL - 100% (1 / 1)

JAVA 12 - 0% (0 / 1)

Test summary

Test	Regular Java Developer - Sabre		
Invitation date	2021-09-22 12:41 GMT	**Test duration**	62 minutes
Test start date	2021-09-27 16:49 GMT	**Test completion date**	2021-09-27 17:52 GMT
Candidate IP	79.184.237.107		

Answers summary

No.	Page title	Duration	Score
#2	Section 1	28 mins 51 secs Suggested: 23 mins	**11** / 20 55.00 %
#3	Section 2	6 mins 5 secs Suggested: 12 mins	**4** / 4 100.00 %
#4	Section 3	5 mins 56 secs Suggested: 7 mins	**3** / 4 75.00 %
#5	Section 4	7 mins 59 secs Suggested: 11 mins	**6** / 7 85.71 %
#6	Section 5	10 mins 19 secs Suggested: 8 mins	**7** / 9 77.78 %
#7	Section 6	3 mins 29 secs Suggested: 6 mins	**4** / 6 66.67 %

Answers details

Section 1

Duration: 28 mins 51 secs (Suggested duration: 23 mins)

Multi choice: Java | Method reference — Score: 2 / 2

In Java 8, having a method declared as below, indicate the correct ways(legal) of invocation with method reference.

```
static void transform(Supplier<Set<String>> strings) {
  //...
```

- ☑ `transform(Collections::emptySet)`
- ☐ `transform(new HashSet())`
- ☑ `transform(HashSet<String>::new)`
- ☑ `transform(HashSet::new)`

Multi choice: Java | Collections — Score: 0 / 2

Is there a class for *LIFO (Last In, First Out)* data processing in Java API?

- ☑ No
- 🟩 Yes

Multi choice: Java | Java8 streams — Score: 0 / 2

What will the following code print (for Java 8)?

```
    import java.util.ArrayList;
    import java.util.List;

    public class Main {

        public static void main(String[] args) {
            String prefix = "Maxi";
            List<String> animals = new ArrayList<>();
            animals.add("Cat");
            animals.add("Dog");
            prefix = "Ultra";
            animals.stream().forEach(animal -> System.out.println(prefix + anima
l));
        }
    }
```

☐ Runtime error

☐
```
MaxiCat
MaxiDog
```

☐
```
Cat
Dog
```

🟩 Compilation error

☑
```
UltraCat
UltraDog
```

Multi choice: Java | The law of Demeter — Score: 2 / 2

In Java, there is *the law of Demeter*, what does it mean?

☑ Code should not contain long calling chains like `objectA.getObjectB().getObjectC().doThat();`

☐ Code should be properly named according to the Java language specification.

☐ Code should be as compact as possible.

Multi choice: Java | Interfaces — Score: 1 / 1

In Java 8, which mechanism is used for preserving backward compatibility when extending the functionality of the interface?

☐ Method reference

☐ Functional interface

☐ Single abstract method

☑ The default method

Multi choice: Java | Optional — Score: 0 / 1

In Java 8, what is the correct way of creating an Optional that may or may not contain a value?

🟩 `Optional.ofNullable(value)`

☐ `Optional.of(null).orElse(Optional.empty())`

☑ `Optional.of(value)`

- ☐ `New Optional(value)`

Multi choice: Java | Try with resources — Score: 0 / 1

Which variable can be placed in a try-with-resource block (since Java 9)?

- ☐ Static
- ☑ Defined outside the block
- 🟩 Final
- 🟩 Effectively final

Multi choice: Java | Strings — Score: 1 / 1

How do you append text to `StringBuffer` object in Java?

- ☐ `insert("teststring")`
- ☑ `append("teststring")`
- ☐ `add("teststring")`

Multi choice: Java | General Terminology - dead code — Score: 1 / 1

What does *dead code* mean?

- ☑ Unreachable code that doesn't generate compilation error.
- ☐ A piece of code causing a program crash.
- ☐ Code in the shutdown hook.
- ☐ Unused local variables, parameters and private methods.

Multi choice: Java | Input Variables — Score: 2 / 2

In the following code, what is the purpose of the three dots?

```
public static String format(String... input);
```

- ☐ input is declared final.
- ☐ input is a list containing exactly three elements.
- ☐ input is a string array containing exactly three elements.
- ☑ input is an array of any number of string elements.

Multi choice: Java | JUnit library Score: 1 / 1

What is *JUnit* library used for?

- [✓] Testing library framework for Java projects.
- [ ] Java web framework library similar to Play or Vaadin.
- [ ] Installation library framework for Java projects.

Multi choice: Java | Collections Score: 0 / 1

Which basic core interfaces are inherited directly from `java.util.Collection`?

- [ ] `Vector`
- [✓] `SortedSet`
- [ ] `Map`
- [✓] `List`
- [✓] `Queue`
- [✓] `Set`

Multi choice: Java | basics - extension Score: 0 / 1

What will the following code print (each class has its own file)?

```
public interface Ship {
    void go();
}
```

```
public class MarineShip implements Ship {
    public void go() {
        System.out.println("Sea surface");
    }
}
```

```
public class Submarine extends MarineShip {
    public void go() {
        System.out.println("Underwater");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        MarineShip submarine = new Submarine();
        ((Submarine)submarine).go();
    }
}
```

- ☐ `Sea surface`
- ☑ Runtime error
- 🟩 `Underwater`
- ☐ Compilation error

Multi choice: Java | False/True — Score: 1 / 1

What will be the result of the following code in Java?

```
public class Main {
    public static void main(String[] args) {
        A a1 = new A();
        A a2 = new A();

        System.out.println(a1 == a2);
        System.out.println(a1.equals(a2));
    }
}
```

Assuming the following `A` class declaration:

```
public class A { }
```

☐
```
false
true
```

☑
```
false
false
```

☐
```
true
true
```

☐
```
true
false
```

Multi choice: Java | Core | Standard API **Score: 0 / 1**

What is the difference between the `equals` method and the `==` operator in Java?

- 🟩 `==` compares references while `equals` compares objects body in the way it has been implemented
- [✓] `equals` compares references while `==` operator compares the objects body
- [ ] There is no difference between them.

Section 2

Duration: 6 mins 5 secs (Suggested duration: 12 mins)

Multi choice: Spock | Test setup **Score: 1 / 1**

The Spock `setup` method is invoked ...

- [ ] Once for each class instance if defined on a class level
- [✓] Before any test method in the class
- [ ] Once before starting the test run in the project

Multi choice: JUnit | Asserts **Score: 1 / 1**

What is the purpose of the following assert statement in JUnit?

```
assertSame([String message], expected, actual)
```

- [✓] Verifies whether the expected `object` is the same as a given `object`.
- [ ] Verifies whether the `object` is empty or not.
- [ ] Verifies whether an expected `object` is different from a given `object`.
- [ ] The code is incorrect.

Multi choice: QA | Inverting the test pyramid **Score: 2 / 2**

Does it make sense to invert the test pyramid?

- [✓] Yes. For example when refactoring legacy software.
- [ ] No, because end-to-end tests are hard to maintain.
- [ ] No. Unit tests always need to be in the majority.

Section 3

Duration: 5 mins 56 secs (Suggested duration: 7 mins)

Multi choice: Git | Basic Git theory — Score: 1 / 1

Which statements about Git are true?

- [✓] Branching is done to allow the user to create their own branch and toggle between those branches.
- [ ] `Git share` command is used to share your work online with others.
- [✓] Git conflicts generally arise when two people have changed the same lines in a file.
- [ ] To share your work with others, you first have to push it and then commit it.

Multi choice: Git | Basic Git theory, part 2 — Score: 1 / 1

Which statements about Git are true?

- [✓] Git is a proprietary software for distributed version control.
- 🟩 Most operations in Git only require local files and resources to operate.
- [✓] A repository is a place where the history of work is stored.
- [✓] Git has three main states that files can reside in, that is committed, modified and staged.

Multi choice: Git | Intermediate Git commands knowledge — Score: 1 / 2

Which statements about Git commands are true?

- 🟩 `Git remote prune` – deletes all stale remote-tracking branches. `Git stash` – stashes changes in a dirty working directory.
- [✓] `Git tag` –allows you to operate with tags. `Git remote` – manages a set of tracked repositories.
- [✓] `Git cherry pick` – applies changes introduced by existing commits. `Git mv` – moves the entire repository into a different location.
- 🟩 `Git bisect` – uses a binary search to find the change that introduced a bug. `Git blame` – shows who changed which lines in a file, and why.

Section 4

Duration: 7 mins 59 secs (Suggested duration: 11 mins)

Multi choice: SQL | Aggregation — Score: 0 / 1

What best describes this query?

```sql
SELECT count(middle_name) FROM customer
```

- [✓] The query counts all the customers that have a (non-NULL) middle name.
- [✓] The query counts all the customers.
- [ ] The query counts all the distinct customer middle names.
- [ ] The query is wrong, we cannot pass a column to the COUNT() aggregate function.

Multi choice: SQL | Aggregation

Score: 1 / 1

What is the best description for this SQL query?

```
SELECT a.first_name, a.last_name, f.title
FROM actor a
LEFT JOIN film_actor fa ON a.actor_id = fa.actor_id
LEFT JOIN film f ON fa.film_id = f.film_id
```

- [ ] The query returns all the actors' names and their films' titles only for actors and films that match each other. Actors without films or films without actors are not returned.
- [ ] There is a syntax error. `LEFT JOIN` does not support `ON`
- [ ] The query returns all the the films' titles, and if any actors played in those films, it also returns their actors' names.
- [✓] The query returns all the actors' names, and if they have played in films, it also returns their films' titles.

Multi choice: SQL | Selecting null values

Score: 1 / 1

We have a `users` table with the following fields: `ID`, `User_Name`, `First_Name`, `Profile`, `Password`, `Country`

Which of the following statements returns all users with `NULL` values in the `Country` field in MySQL?

- [ ] `SELECT * FROM users WHERE Country = 'NULL'`
- [ ] `SELECT * FROM users WHERE Country IS NOT NULL`
- [ ] `SELECT * FROM users WHERE Country <> 'NULL'`
- [✓] `SELECT * FROM users WHERE Country IS NULL`

Multi choice: SQL | Use of COUNT — Score: 2 / 2

What does the following statement do?

```
SELECT COUNT(*) FROM client WHERE name LIKE '%S'
```

- ☐ Gives the number of rows in the `client` table whose `name` contains an `S`.
- ☑ Gives the number of rows in the `client` table whose `name` ends with `S`.
- ☐ Gives the number of rows in the `client` table whose `name` is `%S`.
- ☐ Gives the number of rows in the `client` table whose `name` starts with `S`.

Multi choice: SQL | Ordering — Score: 2 / 2

What will be the order of the resulting dataset for this SQL statement?

```
SELECT continent, population FROM country ORDER BY language ASC, continent ASC
```

`continent` and **`language`** are text fields, while **`population`** is a numeric field.

- ☐ The rows will be in inverse alphabetical order by `continent`. The `language` field won't be taken into account in the ordering because it has not been selected.
- ☐ The rows will be in alphabetical order by `continent`. The `language` field won't be taken into account in the ordering because it has not been selected.
- ☑ The rows will be in alphabetical order by `language`, and for those which have the same `language`, they'll be in alphabetical order by `continent`.
- ☐ The rows will be in inverse alphabetical order by `language`, and for those which have the same `language`, they'll be in inverse alphabetical order by `continent`.

Section 5

Duration: 10 mins 19 secs (Suggested duration: 8 mins)

Multi choice: Java | Spring | Annotations in @SpringBootApplication — Score: 2 / 2

Which annotations are included in **`@SpringBootApplication`**?

- ☑ `@Configuration`
- ☑ `@ComponentScan`
- ☐ `@Repository`
- ☑ `@EnableAutoConfiguration`

Multi choice: Java | Spring | NoUniqueBeanDefinitionException — Score: 0 / 1

How can you resolve a NoUniqueBeanDefinitionException?

- [x] By using @Qualifier annotation.
- [x] By using @Primary annotation.
- [ ] By autowiring the required set to false with flag.
- ■ By autowiring by name.

Multi choice: Java | Spring | AOP   Score: 1 / 1

What does AOP stand for in Spring?

- [ ] Aspect-Oriented Procedure
- [x] Aspect-Oriented Programming
- [ ] Asynchronous Oriented Programming
- [ ] Asynchronous Oriented Procedure

Multi choice: Java | Spring | Default scope   Score: 1 / 1

What is the default scope of a Spring bean?

- [ ] Session
- [x] Singleton
- [ ] Request
- [ ] Prototype

Multi choice: Java | Spring | Dependency Injection   Score: 2 / 2

Which of the following Spring annotations allow resolving dependencies by field injection?

- [x] `@Autowired`
- [ ] `@Required`
- [x] `@Inject`
- [x] `@Resource`
- [ ] `@Component`
- [ ] `@Bean`

Multi choice: Java | Spring | Context hierarchy   Score: 1 / 2

What is true about Application Context Hierarchy?

- ☑ A child context can override the configuration inherited from the parent context.
- ☐ You can not have multiple application contexts which share a parent-child relationship.
- ■ A context can access beans in the parent context but not vice-versa.
- ☐ A context can access beans in the parent context and vice-versa.

Section 6

Duration: 3 mins 29 secs (Suggested duration: 6 mins)

Multi choice: Java | Hibernate | Caching levels — Score: 1 / 1

How many caching levels do we have in Hibernate?

- ☐ 5
- ☐ 4
- ☑ 2
- ☐ 3

Multi choice: Java | Hibernate | Defining the primary key value — Score: 0 / 1

In Hibernate, how do we define the primary key value generation logic as auto?

- ☐ `@Generator(strategy = GenerationType.AUTO)`
- ☑ `@GeneratedValue(strategy = GenerationType.AUTO)`
- ☑ `@Id(strategy = GenerationType.AUTO)`

Multi choice: Java | Hibernate | Fetching ways — Score: 1 / 2

What are the different fetching ways in Hibernate?

- ■ Batch Fetching
- ☑ Join Fetching
- ☑ Select Fetching
- ☑ Sub-select Fetching
- ☐ Inner-join Fetching

Multi choice: Java | Hibernate | Types of Hibernate object states — Score: 2 / 2

What are the three types of Hibernate object states?

- ☐ transparent, persistent, detached
- ☐ transient, persistent, embedded
- [✓ transient, persistent, detached]
- ☐ transient, preserved, detached