**Questions**

**• What's new in Java 8? Explain some of them.**

-Lambda expresions to simplify code and Streams to do array management faster.

**• Given the following list implement a solution in order to get even numbers using Java 8**

**Streams List list = Arrays.asList(1,2,3,4);**

list.stream().filter(numero -> (Integer)numero % 2 == 0).forEach(numero -> System.out.println("El numero: "+numero+" es par"));

**• What do you notice when you do code review?**

I notice if there is “Clean Code” or if i need put comments for explain something.

• **Have you ever worked with Scrum? Tell us what it is, what events do you remember and what roles are involved?**

Yes. I have worked with Scrum. Its a work methodology to develop some proyect or proyects together with “Sprints” (2 or 3 weeks of work to give something to the client).

Everyday i do the Daily. At 9:30 am with all the programmers team. At the beguining of the SPRINT we talk with the Scrum Master and we do that we call a “Retro”. This is talk everybody about the Sprint, what we did wrong and what we did well and tell ourself how can we improve. 1 day every 1 or 2 month, we have a “big” meeting with the Product Owner and more people (Security, Architecture, UX) and we see how it goes the big proyect.

• **What access modifiers (or visibility) do you know in Java?**

Public, private and protected.

**• Differences between an abstract class and an interface**. **When would you use one or the other?**

An abstract class is a class with several methods but one or several of them are specified but not implemented.

And interfaz is a class with ALL the methods inside without implementation.

If i want re-use some functions for several class that do THE SAME, then i use an abstract class for this. But if i only want to say “Every animal in this world can move but in diferents modes”, i would use an interface, because birds move flying, dogs running, fish swimming, etc.

Interfaz? Method Move. Class of implementation? Implement method how every animal moves.

• **What is Maven and why is it used? What is Maven life cycle?**

Maven is a tool that you can use to put all the dependences (local or online) that you want in your project just puting a simple lines on a file (POM).

Its the several steps that Maven do for finally deploy the project. Compile, test, install, deploy.

• **What is Git and what is it used for? List all Git commands that you know.**

Git is a repository online for team work. Several people can work at the same time and implement code together. Commit, Push, Push, Merge, Stash, Rebase and Fetch.

• **What is a mock? What would you use it for?**

A mock is an object that “fake” another object or service, and u can develop your code with this fake object and when the true object will be prepare, u can replace it.

**• How would you explain to someone what Spring is? What can it bring to their projects?**

Spring is a framework to do easier you work. It give you archetype project with the “basics” that you want, like what connection to BBDD you want, or if you want a web module, or just a basic app. Spring can bring to your projects a fastly development to créate the project and deploy it.

• **What's the difference between Spring and Spring Boot?**

Spring is a framework and Spring Boot is a “tool” that use Spring Framework and do it easier and faster to use.

**• Do you know what CQRS is? And Event Sourcing?**

**• Differences between IaaS and PaaS. Do you know any of each type?**

On IaaS you have to do and maintenance everything, include the server. On paaS you cant touch the server of the aplication.

**• Explain what a Service Mesh is? Do you have an example?**

Its the network that connect the microservices.

• **Explain what is TDD? What is triangulation?**

• **Apply the Factory pattern with lambda expressions**

• Reduce the 3 classes (OldWayPaymentStrategy, CashPaymentStrategy and CreditCardStrategy) into a single class (PaymentStrategy). You do not need to create any more classes or interfaces. Also, tell me how you would use PaymentStrategy, i.e. the different payment strategies in the Main class public interface OldWayPaymentStrategy { double pay(double amount); } public class CashPaymentStrategy implements OldWayPaymentStrategy { @Override public double pay(double amount) { double serviceCharge = 5.00; return amount + serviceCharge; } } public class CreditCardStrategy implements OldWayPaymentStrategy { @Override public double pay(double amount) { double serviceCharge = 5.00; double creditCardFee = 10.00; return amount + serviceCharge + creditCardFee; } } public interface PaymentStrategy { //write here your solution } public class Main { public static void main(String[] args) { } }

