1) For java, there are some alternatives frameworks;
   -Salta
   -Dagger
   -Google Guice

   Also for Scala, there is Play Framework, which enables dependency injection.

   For .Net, there are many other options too;
   -Spring.Net
   -Structure Map
   -Castle Windsor

2) @SpringBootApplication annotation is the combine of 3 other annotations;

   -@EnableAutoConfiguration: This lets Spring to create beans and add them to the context automaticly by scanning the classpath.

   -@ComponentScan: This annotation is for the classes and it tells Spring to scan this class to find the beans and make the configurations.

   -@Configuration: Tells Spring this class is a configuration class. Spring then, in addition to component scanning, scans these class and finds the methods with @Bean annotation and add them to the context.

3) @Primary: this annotation gives a higher preference to a particular bean between others. When we scan a class with @Component annotation, we can find multiple beans. If we don't specify the bean's name, then spring goes to the default bean with @Primary annotation.

   @Qualifier: this annotation solves the problem of confliction between same type of beans. When we try to create a bean, if there are multiple beans in same type, we can face with an confliction error. In this case we write @Qualifier to define a specific bean.

4) Conventions over controller is simply following some rules for naming the contoller.

   For This convention; we would use "Controller" at the end of controller class.

   For example;

   WelcomeController maps to the '/welcome*' request URL. So we do put "Controller at the end of name of this class.

5) Aspect oriented programming is a programming paradigm to increase the readibility of code and make it more clean. Also it prevents repetation, decrase the cost of maintain and improvement cost of software. In order to do this, we simply make software for a single pupose. So when we do some other things in the main process (like validation etc), we simply create another piece of software to do this job for us. This way we can focus on the main process in our main software.

6) S-single respoonsibility principle: An object or a function must have one job, one purpose.

   O-open and close principle: Objects and functions must be stable. They should not be changing through time and situation.

   L-liskov substitution principle: we should be able to use our code in the name of derived class without any modification

   I-interface segregation principle: Instead of creating one single interface to decide aout responsibilites, we can create multiple one with more precise definition.

   D-dependency invertion: dependency between classes must be as low as possible.

7) Swagger is an interface description language for describing RESTful APIs expressed using JSON. We use it do visualize our API desgn and with help of Swagger we can make sure that we arre doing the right thing. This testing process can be done in traditional ways too, but this is more accurate and gives an UI to see the bigger picture.

8) This one I did not understand clearly

9) URI: is a formal system to distinguish the differences between resources. These resources consist of 2 types: URL and URN

   URL: also known as web address, defines the resource's location and how to access it. It does not tell us any information about the resource itself.

   URN: this identifies the the resource itself but does not tell us how where is this resource and how to reach it.

10) If we call a function multiple times and if gives us the same result everytime, this means this function is idempotent function. If not then it is not an idempotant function.

    In HTTP REST method, GET, PUT and DELETE are idempotent methods. If we run these methods multiple times, they will not return or create different values after the first time.

However, for the method POST, it is not the same. It will keep increasing the total number of elements in the context. Everytime we call it, we will get a different result.

11) RFC is simply the published documents over internet standarts and protocols. These technical publishes managed by IETF (Internet Engineering Task Force). These are technical details of technologies, and we can simply apply these techniques and implement it. RFC2616 is the one that introduce us HTTP. In this document, it specify almost everything over HTTP. All the response codes, methods like GET etc and it's syntax. It is a document with almost 170 pages.