Transfer service

Solution design and Technology choices

- Since I wanted to explore more on spring-boot so I have chooses it as a base framework for solution.
- For interactions I used REST web services since it's easy to configure and develop.
- For ORM I have used JPA so that the solution is not dependent on implementation and can be changed if needed. Plus CRUD operations are readily available though JPA repository.
- For database I used spring boot readily configured and supported in memory H2 DB.
- I have tried to implement the basic functionalities asked in the problem statement with all expected and supported services
- Wherever necessary I have provided business/format checks/validations and messages.
- For unit test cases I have used Mockito and Hamcrest libraries again readily supported by spring boot.

Limitations:

- I haven't provided any authentication and authorization mechanism since there are many approaches and depends on the practice across organizations. Although basic authentication (user, password) can be activated using spring-boot.
- I also didn't created/handled/reported all types of validation failures/messages uniformly since that require a separate implementation in code and that seems out of scope considering the size of the functionality.
- Same is the case of exceptions like validation/business/generic i.e. no robust catching(at interceptors level too) reporting and resolution

Deployment instructions:

- For using in an IDE it's simple to download the code base and import it as maven project. Clean, build, resolve dependencies and launch as java application with StartApplication.java as starting point
- For deployment on a unix box use following command (make sure to have java8 on path)
  java -jar "TransferService-18.1.0.0.jar"
  Where jar *TransferService-18.1.0.0.jar* is located under *target*\ directory

Generic instructions**:**

- I have included a 'SampleRequest.txt' file with sample requests for invoking available services