

JSON

Lecture Question

Question: In a package named "oop.json" create and complete the "Store" class which is stated below

toJSON returns a JSON string representing an object with keys "cashInRegister" and "inventory" mapping to the values from the two state variables with the same names

fromJSON takes a JSON string in the same format returned from toJSON and sets the state variables to the values from the JSON string

```
package oop.json
```

```
class Store(var cashInRegister: Double, var inventory: List[String]) {
```

```
  def toJSON(): String = {  
    ""  
  }
```

```
  def fromJSON(jsonString: String): Unit = {  
  
  }
```

```
}
```

JSON - Reminder

- JSON is [mostly] used to communicate between programming languages
- Consists of 6 types
 - String
 - Number
 - Boolean
 - Array
 - Object
 - Null

JSON - Reminder

- In Python
 - `json.dumps` to convert from Python types to JSON string
 - `json.loads` to convert from JSON string to Python types
- In JavaScript
 - `JSON.stringify` to convert from JavaScript types to JSON string
 - `JSON.parse` to convert from JSON string to JavaScript types

JSON

- What about Scala?

```
{"timestamp":1550774961,"message":"success","iss_position":  
  {"latitude":"-36.5017","longitude":"-2.8015"}}
```

- This is valid JSON
- What Scala type do we use to store this data?

JSON

- What about Scala?

```
{"timestamp":1550774961,"message":"success","iss_position":  
  {"latitude":"-36.5017","longitude":"-2.8015"}}
```

- This is valid JSON
- What Scala type do we use to store this data?
 - Map[String, String]?
 - Map[String, Long]?
 - Map[String, Map[String, String]]?
 - Map[String, Any]?? <- This is the only one that can work, but it's very restrictive since we can only use the Any methods

JSON

- What about Scala?

```
{"timestamp":1550774961,"message":"success","iss_position":  
  {"latitude":"-36.5017","longitude":"-2.8015"}}
```

- This is valid JSON
- What Scala type do we use to store this data?
 - We can't mix types in our Scala data structures
 - .. at least, not without polymorphism

JSON - Library

- We'll install a library to help us work with JSON in Scala
 - The Play JSON library
- Library defines these Scala types
 - JsString
 - JsNumber
 - JsBoolean
 - JsArray
 - JsObject
 - JsNull
- **All these types extend JsValue**

JSON - Library

- What about Scala?

```
{"timestamp":1550774961,"message":"success","iss_position":  
  {"latitude":"-36.5017","longitude":"-2.8015"}}
```

- This is valid JSON
- What Scala type do we use to store this data?
 - **Map[String, JsValue]**

JSON - Library

- The library parses JSON strings and converts all values into one of the Js_ types
- We store them in variables of the JsValue base class
- Convert values to the Scala types as needed

Reading JSON

JSON - Library

```
response = {"timestamp":1550774961,"message":"success","iss_position":  
            {"latitude":"-36.5017","longitude":"-2.8015"}}
```

```
import play.api.libs.json.{JsValue, Json}
```

```
...
```

```
val parsed: JsValue = Json.parse(response)
```

```
// unused values, but this is how we would extract message and timestamp
```

```
val message: String = (parsed \ "message").as[String]
```

```
val timestamp: Long = (parsed \ "timestamp").as[Long]
```

```
val issLocation: Map[String, String] = (parsed \ "iss_position").as[Map[String, String]]
```

- Use the library to extract specific values

JSON - Library

```
response = {"timestamp":1550774961,"message":"success","iss_position":  
            {"latitude":"-36.5017","longitude":"-2.8015"}}
```

```
import play.api.libs.json.{JsValue, Json}
```

```
...
```

```
val parsed: JsValue = Json.parse(response)
```

```
// unused values, but this is how we would extract message and timestamp
```

```
val message: String = (parsed \ "message").as[String]
```

```
val timestamp: Long = (parsed \ "timestamp").as[Long]
```

```
val issLocation: Map[String, String] = (parsed \ "iss_position").as[Map[String, String]]
```

- Import the classes/objects we'll need from the library

JSON - Library

```
response = {"timestamp":1550774961,"message":"success","iss_position":  
            {"latitude":"-36.5017","longitude":"-2.8015"}}
```

```
import play.api.libs.json.{JsValue, Json}
```

```
...
```

```
val parsed: JsValue = Json.parse(response)
```

```
// unused values, but this is how we would extract message and timestamp
```

```
val message: String = (parsed \ "message").as[String]
```

```
val timestamp: Long = (parsed \ "timestamp").as[Long]
```

```
val issLocation: Map[String, String] = (parsed \ "iss_position").as[Map[String, String]]
```

- Call `Json.parse`
- Parses the JSON string and converts it to a `JsValue`

JSON - Library

```
response = {"timestamp":1550774961,"message":"success","iss_position":  
            {"latitude":"-36.5017","longitude":"-2.8015"}}
```

```
import play.api.libs.json.{JsValue, Json}
```

```
...
```

```
val parsed: JsValue = Json.parse(response)
```

```
// unused values, but this is how we would extract message and timestamp
```

```
val message: String = (parsed \ "message").as[String]
```

```
val timestamp: Long = (parsed \ "timestamp").as[Long]
```

```
val issLocation: Map[String, String] = (parsed \ "iss_position").as[Map[String, String]]
```

- Extract values at specific keys
- Use \ to get the value at a key as a JsValue
- Use as[type] to convert the value to the type you expect
 - Cannot use your custom types without defying how to parse your type

Writing JSON

JSON - Library

```
response = {"timestamp":1550774961,"message":"success","iss_position":  
           {"latitude":"-36.5017","longitude":"-2.8015"}}
```

```
def createJSON(message: String, timestamp: Long, location: Location): String = {  
    val jsonTimestamp: JsValue = Json.toJson(timestamp)  
    val jsonMessage: JsValue = Json.toJson(message)  
  
    val locationMap: Map[String, String] = Map(  
        "latitude" -> location.latitude.toString,  
        "longitude" -> location.longitude.toString  
    )  
  
    val jsonLocation: JsValue = Json.toJson(locationMap)  
  
    val jsonMap: Map[String, JsValue] = Map(  
        "timestamp" -> jsonTimestamp,  
        "message" -> jsonMessage,  
        "iss_position" -> jsonLocation  
    )  
  
    Json.stringify(Json.toJson(jsonMap))  
}
```

- Convert Scala types to JsValue with Json.toJson
- Cannot use your custom types without defining how to convert your type

JSON - Library

```
response = {"timestamp":1550774961,"message":"success","iss_position":  
            {"latitude":"-36.5017","longitude":"-2.8015"}}
```

```
def createJSON(message: String, timestamp: Long, location: Location): String = {  
    val jsonTimestamp: JsValue = Json.toJson(timestamp)  
    val jsonMessage: JsValue = Json.toJson(message)  
  
    val locationMap: Map[String, String] = Map(  
        "latitude" -> location.latitude.toString,  
        "longitude" -> location.longitude.toString  
    )  
  
    val jsonLocation: JsValue = Json.toJson(locationMap)  
  
    val jsonMap: Map[String, JsValue] = Map(  
        "timestamp" -> jsonTimestamp,  
        "message" -> jsonMessage,  
        "iss_position" -> jsonLocation  
    )  
  
    Json.stringify(Json.toJson(jsonMap))  
}
```

- Call `Json.stringify` to convert a type to a JSON string
 - Can be any types known to the library (Most of the common Scala types)

Maven

- We're using a new library
- Must download it before use
- Add it to our Maven file

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0">
  <groupId>com.mycompany.app</groupId>
  <artifactId>test1</artifactId>
  <modelVersion>4.0.0</modelVersion>
  <url>http://maven.apache.org</url>
  <version>0.0.1</version>

  <dependencies>

    <!-- https://mvnrepository.com/artifact/org.scalatest/scalatest -->
    <dependency>
      <groupId>org.scalatest</groupId>
      <artifactId>scalatest_2.12</artifactId>
      <version>3.0.5</version>
    </dependency>

  </dependencies>

</project>
```

- This is our current Maven file that we used to download scalatest
- We can add more dependancies to this file
 - Open the Maven sidebar, refresh, then download the new libraries

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0">
  <groupId>com.mycompany.app</groupId>
  <artifactId>test1</artifactId>
  <modelVersion>4.0.0</modelVersion>
  <url>http://maven.apache.org</url>
  <version>0.0.1</version>

  <dependencies>

    <!-- https://mvnrepository.com/artifact/org.scalatest/scalatest -->
    <dependency>
      <groupId>org.scalatest</groupId>
      <artifactId>scalatest_2.12</artifactId>
      <version>3.0.5</version>
    </dependency>

  </dependencies>

</project>
```

- Find new libraries at <https://mvnrepository.com>
 - An enormous wealth of shared libraries
 - Search for the new libraries, paste the dependency into your pom.xml file

Lecture Question

Question: In a package named "oop.json" create and complete the "Store" class which is stated below

toJSON returns a JSON string representing an object with keys "cashInRegister" and "inventory" mapping to the values from the two state variables with the same names

fromJSON takes a JSON string in the same format returned from toJSON and sets the state variables to the values from the JSON string

```
package oop.json
```

```
class Store(var cashInRegister: Double, var inventory: List[String]) {
```

```
  def toJSON(): String = {  
    ""  
  }
```

```
  def fromJSON(jsonString: String): Unit = {  
  
  }
```

```
}
```

Lecture Question

```
package tests
```

```
import json.Store
```

```
import org.scalatest.FunSuite
```

```
class TestStore extends FunSuite {
```

```
  val EPSILON: Double = 0.000001
```

```
  def equalDoubles(d1: Double, d2: Double): Boolean = {  
    (d1 - d2).abs < EPSILON  
  }
```

```
  test("test the store JSON") {
```

```
    val store: Store = new Store(550.21, List("eggs", "milk", "waffles"))
```

```
    val storeJSON: String = store.toJSON()
```

```
    val store2: Store = new Store(0.0, List())
```

```
    store2.fromJSON(storeJSON)
```

```
    assert(equalDoubles(store2.cashInRegister, 550.21))
```

```
    val actualList: List[String] = store2.inventory.sorted
```

```
    val expectedList: List[String] = List("eggs", "milk", "waffles").sorted
```

```
    assert(actualList == expectedList)
```

```
  }
```

```
}
```