LAB # 5

Intermediate Javascript

OBJECTIVE

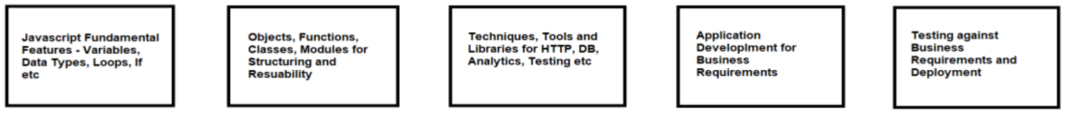
To get familiar and become more knowledgeable in implementing real world JS use cases

THEORY

Javascript is not merely a scripting language anymore. Today JS is used in nearly every type of application, be it a small website, an enterprise application, or a WEB API written in NodeJS. Since applications written today using JS are enterprise based and there are huge business requirements to take care of, it is natural that the JS codebase grows quickly. If some structure is not put to organize and re-use the code and validate, it soon becomes a mess.

Modern JS applications use a wide range of techniques, tools, and libraries to take care of the many requirements, which include, making connection to the Web API using HTTP protocol, persisting data to the database, capturing analytics just to name a few.

Javascript provides many programming constructs to organize and reuse the code which include creating objects, functions, classes, and modules etc. All of them are for structuring and reusing the JS code. Also, when the JS application is developed, we need to test it, so, there is a concept of Unit Testing the code as well to make sure that the code works the way as intended.


Javascript Fundamental Features - Variables, Data Types, Loops, If etc
Objects, Functions, Classes, Modules for Structuring and Resuability
Techniques, Tools and Libraries for HTTP, DB, Analytics, Testing etc
Application Developlment for Business Requirements
Testing against Business Requirements and Deployment

Pic 1 JS Web Application Development

Unlike other classic OOPs languages, *object* can be created in JS without first creating any class. Object is a data structure to collectively hold information. So, it helps put the structure to the code.

Class is another way object can be created just like in other OOPs languages. When a class is created, it promotes reusability but also it permits to have distinct data state as well along with the behavior which is implemented using functions. So, it helps put the structure and provides reusability.

Function is reusability of the behavior only, and interestingly classes in JS is created using *function* in classic JS (ES5 and less).

A *module* helps put the structure and promotes usability of the code. It also provides encapsulation which is nothing but a separation and avoidance of the conflicting data state and behavior.

When it comes to *techniques*, there are few things to be kept in mind. Besides OOPs, there are SOLID principles and DESIGN PATTERNS among others which helps put the structure and also helps in reusability of the code.

There are many *tools and libraries* to be aware of for the most common tasks encountered during web application development which include HTTP interactions, Database persistence, Logging the application state, Testing etc. HTTP library provides features to get and send the data to the backend server. Database library provides features to get and send the data to be persisted. Logging helps log debugging information in case the application malfunction. Testing library provides features to test and validate different JS artifacts including Objects, Functions, HTTP interactions, Database operations, etc.

Testing plays a crucial role in validating business rules that are implemented in JS web application. The main benefit of writing automated tests against the business rules are that they are repeatable as they are not done manually but rather programmatically. This helps cut down the whole time to validate all the requirements if there are changes made in the code.

JS – *NPM packages configuration file*

package.json

```
{
  "name": "lab-4",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "exercise1-transpile": "babel ./exercises/modules/student.js ./exercises/modules/teacher.js
./exercises/modules/student-teacher-assignment.js ./exercises/modules/index.js --out-dir
exercises-output/modules",
    "exercise2-transpile": "babel ./exercises/objects-methods-classes/staff-member-base.js
./exercises/objects-methods-classes/teacher.js ./exercises/objects-methods-classes/hod.js
./exercises/objects-methods-classes/index.js --out-dir exercises-output/objects-methods-classes",
    "exercise3-transpile": "babel ./exercises/http-database-exception-handling-logging/http.js
./exercises/http-database-exception-handling-logging/database.js ./exercises/http-database-
exception-handling-logging/logging.js ./exercises/http-database-exception-handling-
logging/index.js --out-dir exercises-output/http-database-exception-handling-logging",
	"exercise4-transpile": "babel ./exercises/objects-methods-classes/staff-member-base.js
./exercises/objects-methods-classes/teacher.js ./exercises/objects-methods-classes/hod.js
./exercises/testing/index.js --out-dir exercises-output/testing",
    "exercise1": "npm run exercise1-transpile && node ./exercises-output/modules/index.js",
    "exercise2": "npm run exercise2-transpile && node ./exercises-output/objects-methods-
classes/index.js",
    "exercise3": "npm run exercise3-transpile && node ./exercises-output/http-database-
exception-handling-logging/index.js",
	"exercise4": "npm run exercise4-transpile && node ./exercises-output/testing/index.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@babel/cli": "^7.14.8",
    "@babel/core": "^7.14.8",
    "@babel/preset-env": "^7.14.9",
    "axios": "^0.21.1",
    "logging": "^3.3.0",
    "sqlite": "^3.0.6",
    "sqlite3-offline": "^4.3.1"
  },
  "babel": {
    "presets": [
      "@babel/preset-env"
    ]
  },
  "devDependencies": {
    "simple-assert-ok": "^1.0.4"
  }
}
```

JS – *Modules*

student.js

```
let id = 1;
let name = 'student';
let attendClass = function() {
	console.log('attending class');
};

let student = {
	id,
	name,
	attendClass
};

export { student };
```

teacher.js

```
let id = 1;
let name = 'teacher';
let teachClass = function() {
        console.log('teaching students');
};

let teacher = {
        id,
        name,
        teachClass
};

export { teacher };
```

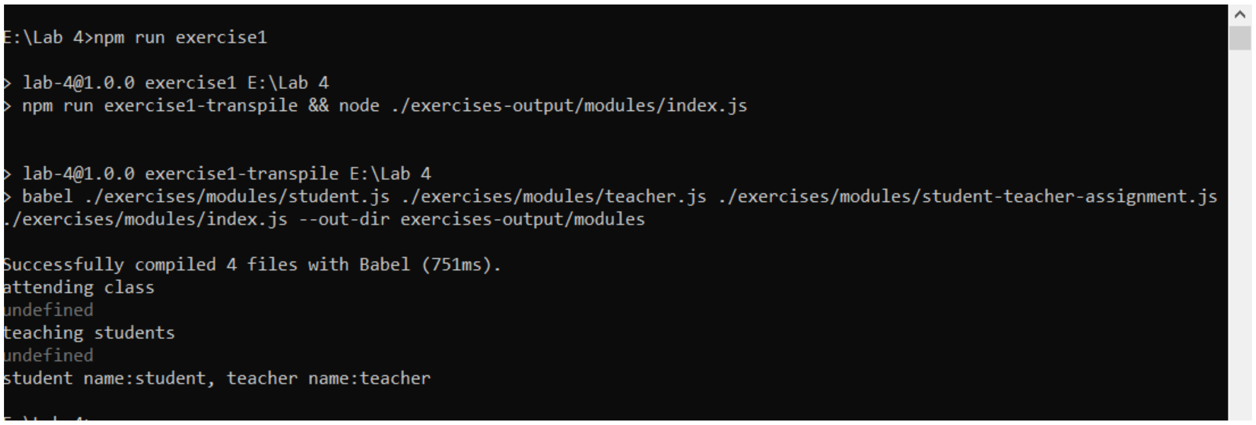
student-teacher-assignment.js

```
import {student} from './student.js';
import {teacher} from './teacher.js';

student.teacher = teacher;

function logAssignment () {
        console.log(student.attendClass());
        console.log(teacher.teachClass());
        console.log(`student name:${student.name}, teacher name:${student.teacher.name}`);
}

export { logAssignment };
```

index.js

```
import {logAssignment} from './student-teacher-assignment.js';

logAssignment();
```

Output



```
E:\Lab 4>npm run exercise1

> lab-4@1.0.0 exercise1 E:\Lab 4
> npm run exercise1-transpile && node ./exercises-output/modules/index.js


> lab-4@1.0.0 exercise1-transpile E:\Lab 4
> babel ./exercises/modules/student.js ./exercises/modules/teacher.js ./exercises/modules/student-teacher-assignment.js
./exercises/modules/index.js --out-dir exercises-output/modules

Successfully compiled 4 files with Babel (751ms).
attending class
undefined
teaching students
undefined
student name:student, teacher name:teacher
```

JS – *Objects, Functions, Classes*

staff-member-base.js

```
class StaffMember {
	constructor(id, name) {
		this.id = id;
		this.name = name;
	}

	visitUniversity(){
		console.log(`${this.name} is visiting university`);
	}
}

export { StaffMember };
```

teacher.js

```
import { StaffMember } from './staff-member-base.js';

class Teacher extends StaffMember {
	constructor(id, name, className) {
		super(id, name);
		this.className = className;
	}

	teachClass(){
		console.log(`${this.name} is teaching ${this.className}`);
	}
}
```

```
export { Teacher };
```

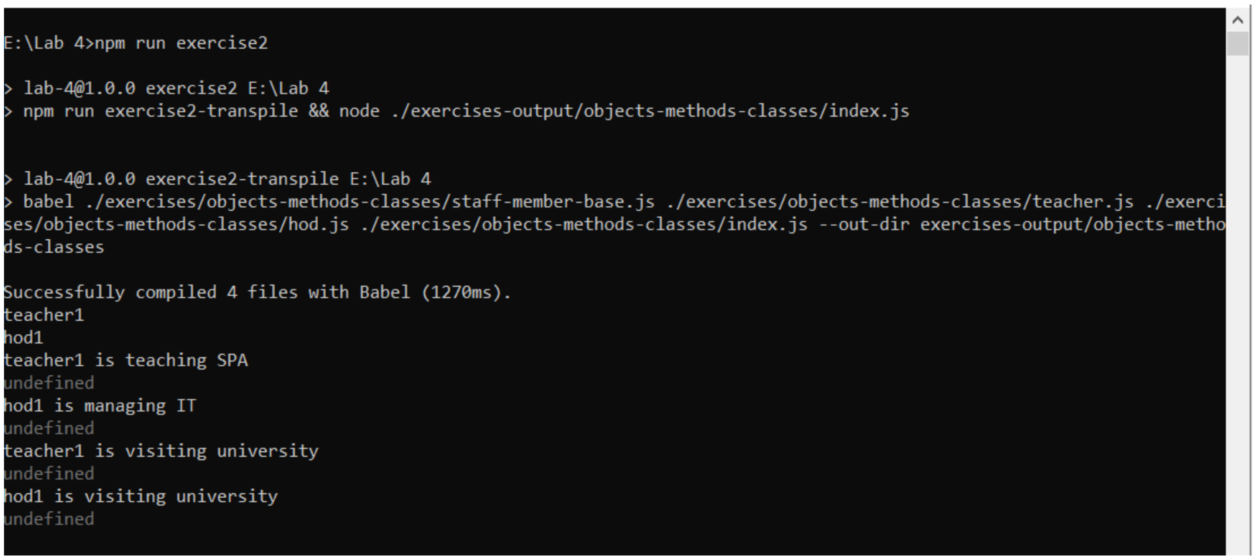
hod.js

```
import { StaffMember } from './staff-member-base.js';

class HOD extends StaffMember {
	constructor(id, name, department) {
		super(id, name);
		this.department = department;
	}

	manageDepartment(){
		console.log(`${this.name} is managing ${this.department}`);
	}
}

export { HOD };
```

indexjs

```
import { Teacher } from './teacher.js';
import { HOD } from './hod.js';

let teacher = new Teacher(1, 'teacher1', 'SPA');
let hod = new HOD(1, 'hod1', 'IT');

console.log(teacher.name);
console.log(hod.name);
console.log(teacher.teachClass());
console.log(hod.manageDepartment());
console.log(teacher.visitUniversity());
console.log(hod.visitUniversity());
```

Output



```
E:\Lab 4>npm run exercise2

> lab-4@1.0.0 exercise2 E:\Lab 4
> npm run exercise2-transpile && node ./exercises-output/objects-methods-classes/index.js


> lab-4@1.0.0 exercise2-transpile E:\Lab 4
> babel ./exercises/objects-methods-classes/staff-member-base.js ./exercises/objects-methods-classes/teacher.js ./exerci
ses/objects-methods-classes/hod.js ./exercises/objects-methods-classes/index.js --out-dir exercises-output/objects-metho
ds-classes

Successfully compiled 4 files with Babel (1270ms).
teacher1
hod1
teacher1 is teaching SPA
undefined
hod1 is managing IT
undefined
teacher1 is visiting university
undefined
hod1 is visiting university
undefined
```

JS – *HTTP, Database, Exception Handling, Logging*

http.js

```
import axios from 'axios';

function getSSUETHomePage(){
	axios.get('https://www.ssuet.edu.pk').then((res) => {
		console.log(res.data);
	});
}

const HTTP = {
	getSSUETHomePage
}

export { HTTP };
```

database.js

```
import { open } from 'sqlite';
import sqlite3Offline from 'sqlite3-offline';

function openStudentDB() {
	console.log('opening DB connection');
	return open('', {
		client: "sqlite",
		connection: {
```

```
                                database: "./db/student.db"
                        }
            });
}

function createStudentTable(db) {
            console.log('creating Student table');
            return db.exec('CREATE TABLE student (id INTEGER, name TEXT)');
}

function insertStudentRecord(db) {
            console.log('inserting Student record');
            return db.exec("INSERT INTO student VALUES (1, 'student1')");
}

function selectStudentRecord(db) {
            console.log('selecting Student record');
            return db.get('SELECT id, name FROM student');
}

const DB = {
            openDB: openStudentDB,
            createTable: createStudentTable,
            insertStudent: insertStudentRecord,
            selectStudent: selectStudentRecord
};

export { DB };
```

logging.js

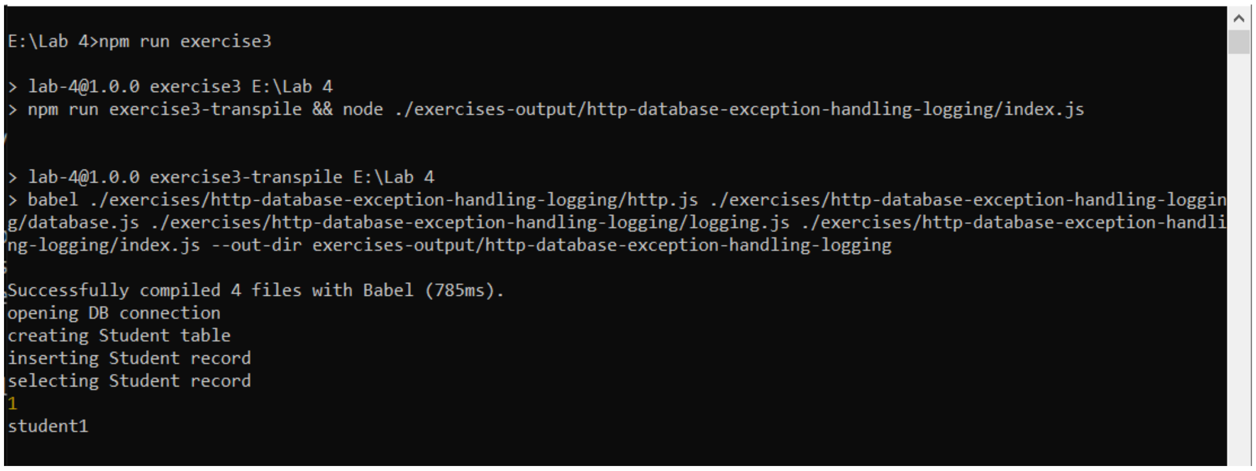
```
import createLogger from 'logging'

let Logger = createLogger('createStudent');

export { Logger };
```

index.js

```
import { HTTP } from './http.js';
import { DB } from './database.js';
import { Logger } from './logging.js';

//Putting the code under try-catch block to avoid crashing the appliation
//and handling the error gracefully
```

```
try {
        //SSUET Home Page - HTTP GET
        getUniversityHomePage();

        //Perform Student DB Operations
        performStudentDBOperations();
} catch (e) {
        //Logging the info in case anything goes wrong
        Logger.debug("Something went wrong");
}

function getUniversityHomePage(){
        HTTP.getSSUETHomePage();
}

function performStudentDBOperations(){
        DB.openDB().then((db) => {
                DB.createTable(db).then(() => {
                        DB.insertStudent(db).then(() => {
                                DB.selectStudent(db).then((student) => {
                                        console.log(student.id);
                                        console.log(student.name);
                                });
                        });
                });
        });
};
```

Output



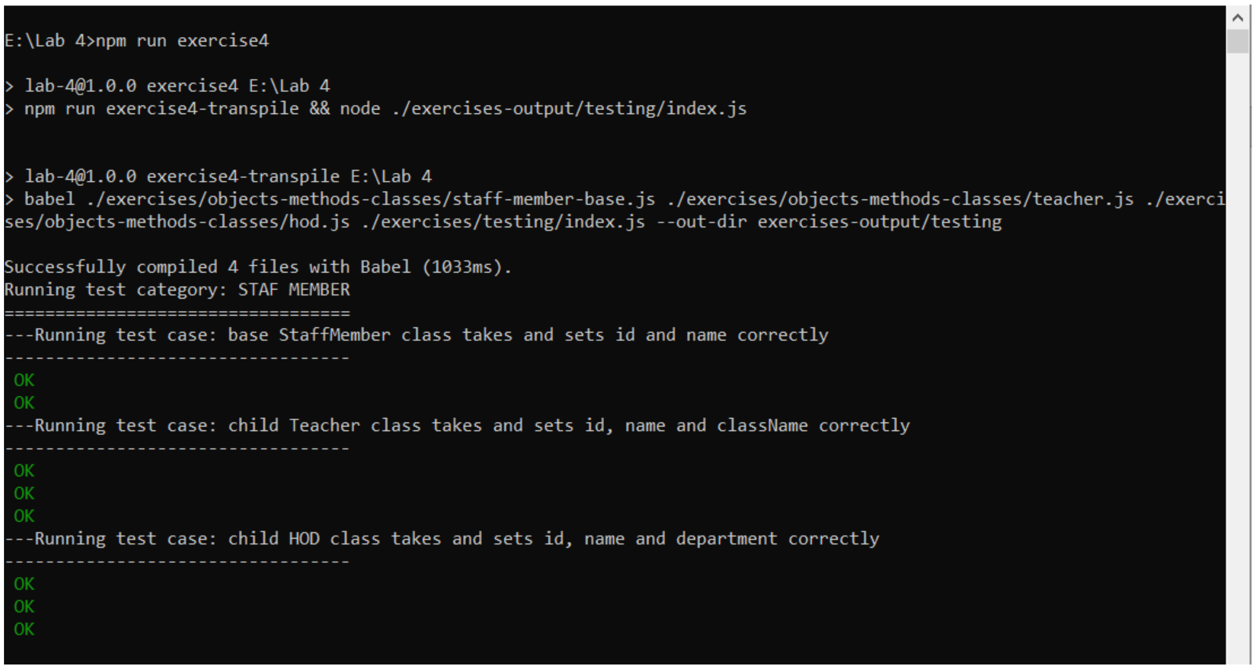
```
E:\Lab 4>npm run exercise3

> lab-4@1.0.0 exercise3 E:\Lab 4
> npm run exercise3-transpile && node ./exercises-output/http-database-exception-handling-logging/index.js


> lab-4@1.0.0 exercise3-transpile E:\Lab 4
> babel ./exercises/http-database-exception-handling-logging/http.js ./exercises/http-database-exception-handling-loggin
g/database.js ./exercises/http-database-exception-handling-logging/logging.js ./exercises/http-database-exception-handli
ng-logging/index.js --out-dir exercises-output/http-database-exception-handling-logging

Successfully compiled 4 files with Babel (785ms).
opening DB connection
creating Student table
inserting Student record
selecting Student record
1
student1
```

JS – Testing

```
import { StaffMember } from './../objects-methods-classes/staff-member-base.js';
import { Teacher } from './../objects-methods-classes/teacher.js';
import { HOD } from './../objects-methods-classes/hod.js';
import assert from 'simple-assert-ok';

let tests = [];

function runTests(){

	describe('STAF MEMBER');

	if(tests.length > 0){
		for(let t = 0; t < tests.length; t++){
			tests[t].run();
		}
	}
}

function describe(category){
	console.log(`Running test category: ${category}`);
	console.log('==================================');

	test('base StaffMember class takes and sets id and name correctly', function() {
		const staffMember = new StaffMember(1, 'member');
		assert(staffMember.id === 1, result('OK'));
		assert(staffMember.name === 'member', result('OK'));
	});

	test('child Teacher class takes and sets id, name and className correctly', function() {
		const teacher = new Teacher(1, 'teacher', 'SPA');
		assert(teacher.id === 1, result('OK'));
		assert(teacher.name === 'teacher', result('OK'));
		assert(teacher.className === 'SPA', result('OK'));
	});

	test('child HOD class takes and sets id, name and department correctly', function() {
		const hod = new HOD(1, 'hod', 'IT');
		assert(hod.id === 1, result('OK'));
		assert(hod.name === 'hod', result('OK'));
		assert(hod.department === 'IT', result('OK'));
	});
}

function test(caseName, func){
	tests = [...tests, {
```

```
                    message: function() {
                              console.log(`---Running test case: ${caseName}`);
                              console.log('---------------------------------');
                    },
                    run: function(){
                              this.message();
                              func();
                    }
          }];
}

function result(testStatus){
          console.log("\x1b[32m", testStatus, "\x1b[37m");
}

runTests();
```

Output



```
E:\Lab 4>npm run exercise4

> lab-4@1.0.0 exercise4 E:\Lab 4
> npm run exercise4-transpile && node ./exercises-output/testing/index.js


> lab-4@1.0.0 exercise4-transpile E:\Lab 4
> babel ./exercises/objects-methods-classes/staff-member-base.js ./exercises/objects-methods-classes/teacher.js ./exerci
ses/objects-methods-classes/hod.js ./exercises/testing/index.js --out-dir exercises-output/testing

Successfully compiled 4 files with Babel (1033ms).
Running test category: STAF MEMBER
==================================
---Running test case: base StaffMember class takes and sets id and name correctly
----------------------------------
 OK
 OK
---Running test case: child Teacher class takes and sets id, name and className correctly
----------------------------------
 OK
 OK
 OK
---Running test case: child HOD class takes and sets id, name and department correctly
----------------------------------
 OK
 OK
 OK
```

Lab Task

1 Classes and Inheritance:
   a. Create a base class called *Student* and export it
      i. Add id, name, date of birth properties
      ii. Add *enroll* method
   b. Create two child classes called *RegularStudent* and *ExecutiveStudent* and inherit them from *Student* base class
      i. Add *attendLab* method in *RegularStudent* class
      ii. Add *attendTheory* method in *ExecutiveStudent* class
   c. Create a separate module and import *RegularStudent* and *ExecutiveStudent* classes. Validate using *console.log* that the properties from base classes have been inherited into child classes along with own properties.

2 HTTP and Database:
   a. Import *sqlite* and *axios* packages in separate module files
   b. Create DB, open connection and perform CRUD operations for *Student using sqlite* library
      i. *Create* and open database
      ii. *Create* Student table using properties from *Student* class created above
      iii. *Insert* two records for both regular and executive student
      iv. *Select* all records
   c. Fetch the university *About* page using HTTP *axios* library
      i. Using *GET* method of axios library, fetch SSUET website *About* page

3 Testing:
   a. Test *Student* class
      i. Use the example above to copy paste and validate Student properties
   b. Test either *RegularStudent* or *ExecutiveStudent* class
      i. Use the example above to copy paste and validate either *RegularStudent* or *ExecutiveStudent* class.

Home Task

1 Classes and Inheritance:
   a. Create a class called *Course* and export it
      i. Add id, name, credit hours properties
   b. Create a class called *University* and export it
      i. Add name, image properties
      ii. Add a method called *setImage*
   c. Modify the *Student* class
      i. Add a property called *university* of type *University* class
      ii. Add a property called *courses* of an array of *Course* class
      iii. Add a method called *addCourse* accepting *Course* class object and set that onto *courses* property
      iv. Add a method called *belongsToUniversity* accepting *University* class object and set that onto *university* property

2 HTTP and Database:
   a. Modify the University class's *setImage* method

        i. Fetch the SSUET logo from the SSUET website using *axios GET* method and set it to *image* property
        ii. Create a new instance and call *setImage* method on the instance and validate using console.log if the image property has been set

    b. Create DB, open connection and perform CRUD operations for *University* using *sqlite* library
        i. *Create* and open database
        ii. *Create* University table using properties from *University* class created above
        iii. *Create* two *University* instances
        iv. *Call setImage* method on both the instances
        v. *Insert* both the instances using their *name* and *image* properties

3 Testing:
    a. Test *University* class.
        i. Create a *University* instance
        ii. Assert if the instance has image property set to empty string
        iii. Call *setImage* method on the new instance
        iv. Assert if the instance has image property set to some string