ITCS-2830 Applications Implementation & Testing

Project #5 – Creating JUnit Unit Tests & TDD

Objectives

Now that you have a solid grasp on **Eclipse, Java and JUnit**, this project will introduce you to writing ***your own JUnit unit tests given a Java class***. You will also try your hand at TDD.

Overview

In this project you are going to create JUnit unit tests for several classes. You will be provided the class but, it is up to you to design solid JUnit **passing** tests. You will not be given the unit tests. It is up to you to design a solid testing solution.

Project 5 – Part I

You are currently developing a system for a local hospital that will manage the new born nursery center. You were given a class called "**Baby.java**" that represents a new born baby. *Please see the Appendix for the **Baby.java** source code.* When you develop the unit tests, try your hand at TDD by creating the unit tests first, running the failing test, developing the class code, and then creating the passing test. Please see the next page for the TDD flow chart.

Your task is to design, implement, and execute passing JUnit unit tests for the Baby.java class. Create a new project in Eclipse called "**Project 5 – Part I**" and copy the code to your source directory **src**. Create a new directory called "**test**" to hold your JUnit unit tests.

A. Create at least **two (2)** Baby objects and test their methods.

B. Use the *appropriate assert* methods to test your class. [Note: you may want to reference previous videos you watched for hints and tips on how to write those tests.]

C. You can assume that the weight and height will never have a zero value (assume the data validation is handled outside of this class before it is used).

Project 5 – Part II

In this project, you are going to test a class that another developer has written called **BankAccount.java**. *Please see the Appendix for the **BankAccount.java** source code.*

Banking applications require a high degree of accuracy and data integrity so you have been tasked with writing JUnit unit tests for this class. Your job is to design, implement, and execute passing JUnit tests for the **BankAccount.java** class. Copy the source code **BankAccount.java** to a new Eclipse project called "**Project 5 – Part II**". Create a new directory called "**test**" to hold your JUnit tests. When you develop the unit tests, try your hand at TDD by creating the unit tests

ITCS-2830 Applications Implementation & Testing

Project #5 – Creating JUnit Unit Tests & TDD

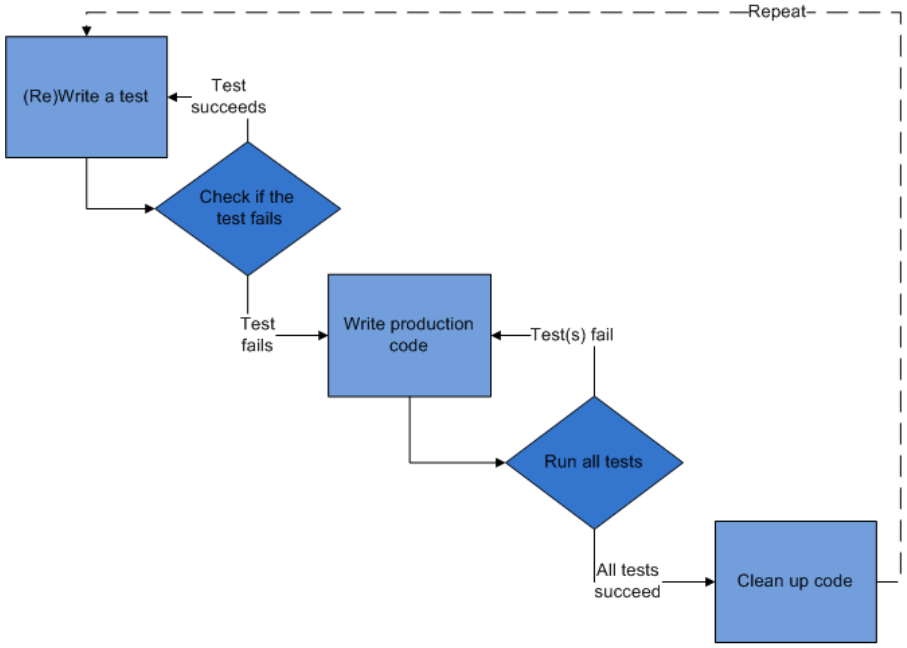
first, running the failing test, developing the class code, and then creating the passing test. Please see the next page for the TDD flow chart.

A. Create at least **two (2)** BankAccount objects and test their methods.

B. Use the *appropriate assert* methods to test your class.

C. Make sure you create objects of the class with an "overdraw" on the account. In other words, if you only have $100.00 in the account test for a withdrawal of over that amount.

Test Driven Development Flow Chart



Test Driven Development Flow Chart

You are now complete with Project #5.

[Continued on next page…]

ITCS-2830 Applications Implementation & Testing

Project #5 – Creating JUnit Unit Tests & TDD

Deliverables

Locate the workspace for your project.

(1) Take a screenshot of your PASSING JUnit tests for Part I and

(2) Take a screenshot of your PASSING JUnit tests for Part II

(3) Zip your entire Project5. A total of THREE separate files (2 - .png screenshots, and 1 – zipped Eclipse project) (no .rar files please, only .zip)

Check List

1 - .png screenshot of Part I……. ☐

2 - .png screenshot of Part II……. ☐

3 - .zip file of your Eclipse project…… ☐

Upload your **THREE** zipped project *files* to the dropbox for **Project #5**.

Please do not zip the screenshots with the zipped Eclipse project – you will have **three** files to upload into the dropbox.

Appendix

```
public class Baby {

	private String firstName, lastName, middleName, deliveryDoctorLastName, bloodType;
	private int dayOfBirth, monthOfBirth, yearOfBirth;
	private double weightInPounds, lengthInInches;
	private boolean discharged = false;

	public Baby(String firstName, String lastName,
				String middleName, String deliveryDoctor,
				double weightInPounds, double lengthInInches,
				String bloodType, int dayOfBirth,
				int monthOfBirth, int yearOfBirth,
				boolean discharged) {

		this.firstName = firstName;
		this.lastName = lastName;
		this.middleName = middleName;
		this.deliveryDoctorLastName = deliveryDoctor;
		this.bloodType = bloodType;
		this.weightInPounds = weightInPounds;
		this.lengthInInches = lengthInInches;
		this.dayOfBirth = dayOfBirth;
		this.monthOfBirth = monthOfBirth;
		this.yearOfBirth = yearOfBirth;
		this.discharged = discharged;
	}

	public String getFirstName() {
		return firstName;
	}

	public String getLastName() {
		return lastName;
	}

	public String getMiddleName() {
		return middleName;
	}

	public String getDeliveryDoctor() {
		return deliveryDoctorLastName;
	}

	public double getWeightInPounds() {
		return weightInPounds;
	}

	public double getLengthInInches() {
		return lengthInInches;
	}

	public String getBloodType() {
		return bloodType;
	}

	public int getDayOfBirth() {
		return dayOfBirth;
	}

	public int getMonthOfBirth() {
		return monthOfBirth;
	}

	public int getYearOfBirth() {
		return yearOfBirth;
	}

	public boolean getDischargedStatus() {
		return discharged;
	}

	public void setDischargedStatus(boolean value){
		this.discharged = value;
	}

	public String toString(){
		return this.firstName+ " " + this.middleName + " " + this.lastName;
	}
}
```

```
public class BankAccount {
	private double balance;
	private String accountNumber;
	private boolean isAccountActive;

	public BankAccount(double balance, String accountNumber) {
		this.balance = balance;
		this.accountNumber = accountNumber;
	}

	public double getBalance() {
		return balance;
	}

	public void setBalance(double balance) {
		this.balance = balance;
	}

	public String getAccountNumber() {
		return accountNumber;
	}

	public void setAccountNumber(String accountNumber) {
		this.accountNumber = accountNumber;
	}

	public boolean isAccountActive() {
		return isAccountActive;
	}

	public void setAccountActive(boolean accountActive) {
		this.isAccountActive = accountActive;
	}

	public void deposit (double amount) {
	 if( amount > 0) {
			balance += amount;
		}
	}

	public void withdrawal(double amount ) {
		if (balance >= amount) {
			balance -= amount;
		}
	}

	public String toString() {
		String msg = String.format ("%s%s  %s$%.2f", "Account Number: ",
accountNumber,
                                                                "Balance: ", balance);
		return msg;
	}
}
```