AT2 – Practical (Java II)

Qualification details			
Qualification National Code & Title	ICT40518 Certificate IV in Programming ICT50718 Diploma of Software Development	**State code**	BEH5 BEG8

Assessment Title *(as per DAP)*	Practical		
Unit National Code & Title	ICTPRG503 Debug and monitor applications ICTPRG527 Apply intermediate object-oriented language skills	**State code**	AUV76 AUV94
Date Due	Refer to the DAP	**Date Received**	

Student Name & ID	
Student Declaration	I declare that the evidence submitted is my own work: ……

Assessor Name	Ken Beck			
Assessment Decision	☐ Satisfactory		☐ Not Yet Satisfactory	
Is student eligible for reassessment (Re-sit)?	☐ Yes	☐ No	**Reassessment Date**	
Assessor Signature			**Date**	

Feedback to student
Feedback from student

Student signature		**Date**	

Candidate Instructions	
The analysis, design, coding, testing and project documentation of a Java application as described on the following page.	
Duration of Assessment	Week 7 ~ 9
Location of Assessment	In class or own place
Reasonable adjustment	In some circumstances, adjustments to assessments may be made for you. If you require support for literacy and numeracy issues; support for hearing, sight or mobility issues; change to assessment times/venues; use of special or adaptive technology; considerations relating to age, gender and cultural beliefs; format of assessment materials; or presence of a scribe you need to inform your lecturer.

Resources Required
Windows 10 system environment / Java SDK software package and NetBeans IDE / Oracle Academy Practice files / Blackboard / Internet

Performance Measurement
Practical based questions that should be completed after the last session and submitted by the due specified in the DAP.

Assessor Instructions	
Type of Assessment	Weekly practice exercises and programming design
Duration of Assessment	Week 7 ~ 9
Location of Assessment	In class or own place
Conditions	Student works are submitted to Blackboard and feedback is given to student on Blackboard
Marking Checklist	Refer to the Marking Guide
Due Date	Refer to the DAP

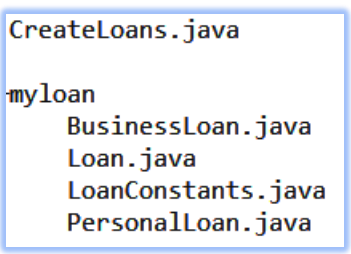
Assessment Specification

NOTE:

- There are different types of questions
  - If the question requires performing some processes, screenshots are required.
  - If programming is required, make sure you include your source files (xxx.java) and provide screenshots of programs running.
    - Your programs cannot be assessed without your source files.
  - If answering a question is required, write your answers in MS-WORD files.
  - With all other types of questions, save your answers appropriately.
  - All programming code must comply with the Java Code Convention.
    - Auto Format option is available if NetBeans is used.
- Zip all your files into a single .zip file before submit.
  - Save your zip file as the submit name, e.g. Portfolio_Week1.zip.
- **Cheating and plagiarism may result in unit re-enrol.**

Activity 1

- ABC Loan Co. makes loans for construction projects. There are two categories of Loans—those to businesses and those to individual applicants.
  - Write an application that tracks all new construction loans. The application must also calculate the total amount owed at the due date (original loan amount + loan fee). The application should include the following classes:
    - ***LoanConstants*** — A public interface. ***LoanConstants*** includes constant values for short-term (1 year), medium-term (3 years), and long-term (5 years) loans. It also contains constants for the company name and the maximum each loan amount ($250,000). Save your file as **LoanConstants.java**.
    - ***Loan*** — A public abstract class that implements the ***LoanConstants*** interface. A Loan includes a loan number, customer last name, amount of loan, interest rate, and term. The constructor requires data for each of the fields except interest rate. Do not allow a loan amounts over the maximum specified in the **LoanConatants** interface. Force any loan term that is not one of the three defined in the LoanConstants class to a short-term, one-year loan. Override the toString() method to display the loan data. Save your file as **Loan.java**.
    - ***BusinessLoan*** — A public class that extends Loan. The ***BusinessLoan*** constructor sets the interest rate to 1% over the current prime interest rate. Save your file as **BusinessLoan.java**.
    - ***PersonalLoan*** — A public class that extends Loan. The ***PersonalLoan*** constructor sets the interest rate to 2% over the current prime interest rate. Save your file as **PersonalLoan.java**.
    - ***CreateLoans*** — An application that creates an **ArrayList** object that contains five ***Loans***. Prompt the user for the current prime interest rate. Then, in a loop, prompt the user for a loan type and all relevant information for that loan. Store the created Loan objects in the array. When data entry is complete, store the **ArrayList** object in a binary file and display all the loans. Save your file as **CreateLoans.java**.
  - The four java files – ***Loan. LoanConstants, BusinessLoan. PersonalLoan*** – must be in the package ***myloan***. The test application will be ***CreateLoans*** in the default package (top folder). The structure of files and package(folder) is as the image below



- Review your program against the program requirements and identify possible runtime errors, and improve program stability by using exception-handling techniques
- Create program API documentation for the classes used in this program.
  - Use *JavaDoc* utility.
- Save your files correctly as specified above.
- Include screenshots of demonstration.

Activity 2

- Write a program that contains 10 numbers in an ArrayList object and let the user search a number.
  - This program initially shows the 10 numbers sorted in ascending order and let the user enter a number for search.
  - It will display if the user entered number is found from the 10 numbers in the **ArrayList** object.
  - Save your file as **ArraySearch.java**.
  - Include screenshots of demonstration.

Activity 3

- Write a Java GUI program that searches and displays the name of an ID from a database table.
  - Create a database table that has two fields, ID and Name. Add 10 records in the table.
    - Provide your SQL statements for database/table creation in a text file (e.g. sql.txt).
    - **NOTE**: The assessor will copy your SQL statements (exactly as it is) and run in the MySQL command to create the database/table/records you used in your program. All the database records required in your program must be correctly created. Test your SQL statements before submit.
  - The program will let the user enter an ID and it will display the name of the ID. Implement the search with SQL statement, e.g. SELECT statement.
    - Provide a textfield for user input.
    - The result (name) is displayed on another GUI component, e.g. another textfield.
  - Keep record of debugging process as you develop this program. Provide a screenshot as an evidence of this process.
  - Write a test documentation of this program. Include sample user inputs and the results.
  - Assess the performance of your program with NetBeans Profiler. Provide a screenshot as an evidence of this process.
  - Save your program as **IDNameDatabase.java**.
  - Include screenshots of demonstration.

End of Assessment Tool