| Document Details | | | |
|---|---|---|---|
| **Version Number** | **Last Updated** | **Developed/Edited By** | **Validation Date** |
| 1.5 | 20/09/2018 | Ken Beck | Dec 2018 |

| | |
|---|---|
| **Training Package** | ICT Information and Communications Technology Training Package |
| **Qualification Title** | ICT50715 AWE6 Diploma of Software Development |
| **Cluster Title** | Java II |
| **Assessment Title** | Portfolio |

**Brief Description of Assessment Task**

Portfolio of 5 questions that should be completed at the end of each session.

**Units of competency, elements, performance criteria to be assessed:**

| State ID | National ID | Competency Title | Elements |
|---|---|---|---|
| AUV76 | ICTPRG503 | Debug and monitor applications | All |
| AUV94 | ICTPRG527 | Apply intermediate object-oriented language skills | All |

| | | | |
|---|---|---|---|
| **Date of Assessment** | Weekly. Week 1 ~ 6 | **Completed by** | Week 6 |
| **Instructions to Students** | The analysis, design, coding, testing and project documentation of a Java application as described on the following page. | | |
| **Resources Required** | Java SDK and IDE / Oracle Academy Practice files / Blackboard / Internet | | |
| **Instructions to Lecturer/Assessor** | Collect and assess each portfolio question at the end of each session. | | |
| **Lecturer's Details** | Name: Ken Beck<br>Email: Ken.Beck@smtafe.wa.edu.au<br>Location: Thornlie Campus | | |

*Students to sign this document when submitting an assessment*

| | |
|---|---|
| **Date Submitted** | |
| **Student Declaration:** | • I have read and understand the details of the assessment.<br>• I have been informed of the conditions of the assessment and the appeals process.<br>• I agree to participate in this assessment.<br>• I certify that the attached is my own work. |

| Student ID | Student Name | Student Signature |
|---|---|---|
| | | |

| Assessment Feedback (Lecturer and Student Copy) | | | |
|---|---|---|---|
| **Assessment Title** | Portfolio | | |
| **Candidate name** | | **Attempt No** | |
| **Assessor name** | | | |
| Performance demonstrated by this assessment is | Satisfactory ☐ | Not Yet Satisfactory ☐ | |
| Assessment outcome and feedback received on | Date: | | |
| **Assessor Comments:** | | | |
| | | | |

| | | | |
|---|---|---|---|
| **Candidate signature:**<br>*(once feedback has been received)* | | **Date** | |
| **Assessor signature:**<br>*(once feedback has been provided)* | | **Date** | |

**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**

- Do the task below

**Date Night at the Arcade**

**Overview**

Tonight is date night at the arcade. After great evening of playing games and winning prizes, you and your date can't help wondering "How are these machines programmed?". You discuss possible designs on the subway back to campus. You enjoy the rest of the night romantically programming your ideas together.

You've made several observations about the arcade. A terminal is used to convert money into game credits. Credits are loaded onto plastic game cards. This data is stored in a card's magnetic strip. Cards may be swiped at any arcade game through the game's magnetic card reader. Games subtract credits from a card, but awards tickets. Tickets are also stored on a card's magnetic strip. Tickets may be exchanged for prizes at the terminal. The terminal is also used to check a card's credit balance and ticket count, and to transfer credits or tickets between cards.

**Tasks**

Write a Java program that models the properties, behaviours, and interactions of objects at the arcade. You'll also need a test class that contains a main method. Use the main method to model actions that would drive the program such as object instantiations and card swipes. All fields must be private. Provide getter and any necessary setter methods.

**Cards**

The magnetic strip on game cards offers limited storage space and zero computing power. Cards store information about their current credit balance, ticket balance, and card number. Neither balance should ever be negative. Individual cards are incapable of performing calculations, including simple addition, or realizing that their balances could go negative.

Every card is created with a unique integer identification number. Although each individual card is incapable of simple addition, it's still possible to perform calculations with properties that belong to all cards.

**Games**

Games require a certain number of credits to be played. Each game is equipped with a magnetic card reader and LCD display. Swiping a card reduces its credit balance, but awards a random, non-negative number of tickets. Print the card number, number of tickets won, along with the new total. Print a message if a card has insufficient credits to play a game.

The "Win Random Tickets Game!" is actually a terrible game. You're welcome to create something more complex, but it's not necessary for this assignment.

**Prize Categories**
Each prize category has a name, number of tickets required to earn that prize, and a count of how many items of this category remain in a terminal. Prizes know nothing about the terminal they belong to.

**Terminals**
Each terminal contains a magnetic card reader. A terminal accepts money which is converted to credits on a card. Money is accepted as whole numbers. Credits are awarded at a rate of 2 credits for every $1. Players may use a Terminal to check their card's balances. Include the card's number in this printout. All or just a portion of credits or tickets may be transferred between cards. Always print a card's balances when either credits or tickets are accessed through a terminal. Finally, tickets may be exchanged at terminals for prizes. Print an error message if a card has insufficient tickets or if the terminal is out of a particular prize type. Print when a prize is awarded and the remaining number of that prize type in the terminal. A terminal offers 3 categories of prizes.

**Main Method**
Instantiate 2 cards and whatever other objects might be necessary to test your program.

- Load credits onto each card.
- Play a bunch of game using both cards.
- Transfer the balance of credits and tickets from Card 1 to Card 2.
- Request prizes using Card 2.
- Try to play a game and request a prize using Card 1.
- Perform whatever other actions might be necessary to test your program.

- Create a program that demonstrates multiple level inherited classes, at least 3 levels, e.g. Animal <- Mammal <- Tiger. Include attributes and methods in each class.

**Activity 2**

- Do the task below

**Overview**
It's been a brutally cold and snowy winter. None of your friends have wanted to play soccer. But now that spring has arrived, another season of the league can begin. Your challenge is to write a program that models a soccer league and keeps track of the season's statistics.

There are 4 teams in the league. Matchups are determined at random. 2 games are played every Tuesday, which allows every team to participate weekly. There is no set number of games per season. The season continues until winter arrives.

The league is very temperature-sensitive. Defences are sluggish on hot days. Hotter days allow for the possibility of more goals during a game. If the temperature is freezing, no games are played that week. If there are 3 consecutive weeks of freezing temperatures, then winter has arrived and the season is over.

**Tasks**
Write a program that models a soccer league and keeps track of the season's statistics. Carefully consider what data should be stored in an array and what data should be stored in an ***ArrayList***. Design classes with fields and methods based on the description of the league. You'll also need a test class that contains a main method. All fields must be ***private***. Provide any necessary getters and setters.

**Teams**

Each team has a name. The program should also keep track of each team's win-total, loss-total, tie-total, total goals scored, and total goals allowed. Create an array of teams that the scheduler will manage.

Print each team's statistics when the season ends.

**Games**
In a game, it's important to note each team's name, each team's score, and the temperature that day. Number each game with integer ID number. This number increases as each game is played. Keep track of every game played this season. This class stores an ***ArrayList*** of all games as a field.

Your program should determine scores at random. The maximum number of goals any one team can score should increase proportionally with the temperature. But make sure these numbers are somewhat reasonable.

When the season ends, print the statistics of each game. Print the hottest temperature and average temperature for the season.

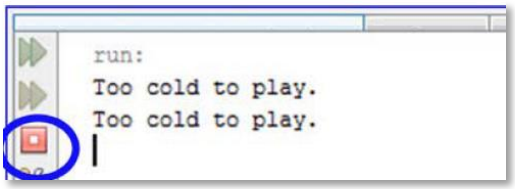
**Scheduler**
Accept user input through a ***JOptionPane*** or ***Scanner***. While the application is running, ask the user to input a temperature. The program should not crash because of user input. If it's warm enough to play, schedule 2 games. Opponents are chosen at random. Make sure teams aren't scheduled to play against themselves. If there are 3 consecutive weeks of freezing temperatures, the season is over.

**Sample Output:**

```
run:
Too cold to play.
Too cold to play.
Too cold to play.
Season is over

*********RESULTS*********

Team 1
Wins: 1, Losses: 1, Ties:0
Points Scored: 9, Points Allowed: 9

Team 2
Wins: 1, Losses: 1, Ties:0
Points Scored: 8, Points Allowed: 8

Team 3
Wins: 0, Losses: 1, Ties:1
Points Scored: 6, Points Allowed: 9

Team 4
Wins: 1, Losses: 0, Ties:1
Points Scored: 8, Points Allowed: 5

Game #1
Temperature: 90
Away Team: Team 2, 4
Home Team: Team 4, 7

Game #2
Temperature: 90
Away Team: Team 1, 8
Home Team: Team 3, 5

Game #3
Temperature: 35
Away Team: Team 1, 1
Home Team: Team 2, 4

Game #4
Temperature: 35
Away Team: Team 3, 1
Home Team: Team 4, 1
```

```
Hottest Temp: 90
Average Temp:62.5
```

**NOTE:**
It's possible that your program may get stuck in an infinite loop during this exercise. This isn't uncommon during software development. The final program should have a way of terminating itself. But until you've implemented that feature, NetBeans provides a ***Stop*** button.




**Exception Handling**
Identify possible runtime errors in the program you've just completed, and improve program stability by using exception-handling techniques.

**Activity 3**

- Create an interactive GUI Java program using JavaFX.
  - Your program must include at least one control that interacts with the user, e.g. button, check box and radio button.
    - An example of user interaction: the button click changes the text in a label.

**Activity 4**

- Create a Java program that reads and update records from a database table and displays them on the screen
  - Use a simple MySQL table with two columns: subject and score
    - Add a few records in the table, e.g. English:95, Math:98, Science:89

**Activity 5**

- Create a java program that contains Car objects in a collection type, e.g. ***ArrayList***. You must demonstrate below.
  - A sample Car class code is provided below. You will need to modify this to meet the activity requirements.
  - Put ***Car.java*** in a separate package from your main class.
  - Store several ***Car*** objects in a ***ArrayList*** object
  - Your program sorts the cars by its make with the ***Collections.sort()*** method
  - Your program has a search method that searches cars with a keyword of car make
  - Your program stores the collection object in a binary file. Use the ***Seriallizable*** interface
  - Your program reads the object from the binary file and display the contents on the screen
  - Show how you used the NetBeans Debugger and Profiler for software development.
    - Describe how the tools help process of development and increase the productivity. Write a test procedure and test document for test execution. Provide screenshots.
  - Provide the API document of your program.
  - ***Car.java*** code: Please modify this sample code as needed for this activity

```
public class Car {

    private String make;
    private String model;
    private String year;
    private int odometer;
```

```
    public Car(String make) {
        this.make = make;
    }

    public Car(String make, String model, String year, int odometer) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.odometer = odometer;
    }

    public void display() {
        System.out.println(" " + make + " " + model + "\t\tYear: " + year);
        System.out.println("\t\t\tkm travelled: " + odometer);
    }

    public String getMake() {
        return make;
    }
}
```

**End of Assessment Tool**