2022

DART Lab - Student Worksheet

ITSE-2123: Advanced Mobile Programming

Enjoy Dart by practicing. Dart is a programming language designed for client development, such as for the web and mobile apps. It is developed by Google and can also be used to build server and desktop applications.

Yoseph Abate
School of Information Technology and Engineering, Addis Ababa Institute of Technology, Addis Ababa University
9/4/2022

Dart Lab 0: Hello World

1. Navigate to the `c:\dart` folder on your computer and create a folder to store the work you will do in this class. The name of the folder should have ***no*** spaces and include part of your name or a nickname, so you will recognize it as yours in future classes.

2. Open the folder in the command prompt and run the following command to create a basic console appliation

```
dart create dart_oop_programming
```

3. Change the directory to the newly created project.

```
cd dart_oop_programming/
```

4. Open the project in VSCode

```
code .
```

5. Open the file `bin/dart_oop_programming.dart` file in VSCode.

```
void main(List<String> arguments) {
  print('Hello Dart!');
}
```

6. Click on the "Run and Debug" button on the left pane of VSCode. Click on the "create json file". This process shall create the file "`.vscode/lunch.json`" in the project. Run the file "`dart_oop_programming.dart`".

7. Now change the `dart_oop_programming.dart` code so that it prints out "Goodbye, World!" instead. This should be done by changing only *one* line of your program. Compile and run your program and see what it prints out.

8. The command `print()` prints out its argument and then starts a new line. Change your program so it prints out "Hello, Dart!" on one line and then prints out "Goodbye, World!" on the next line. Compile and run.

9. Take a look at the code you have written. After the word `main()` in your code, there is an opening brace `{` which denotes the beginning of the class. Then all the way at the bottom of your code there is a closing brace `}` which denotes the end of the function.

10. Every Dart file ends with the extension of “`dart`”. You can write different classes within the same file.

12. Add these lines to your main method:

```
    String name = "AAiT";
    print("Hello,");
    print(name);
    print("How are you today?");
```

Compile and run.

11. Change the text "AAiT" to your name (for example, "Halima") and compile and run your code again. How has the output changed?

12. Change the line

```
    print(name)
```

to the line

```
    print("name");
```

Why are the outputs different?

Dart Lab 1: Variables & Operators

1. Correct the following statements:

```
  a) bool isGood = 1;
  b) String firstLetter = p;
  c) int 2way = 89;
  d) String name = Manish;
  e) int player score = 8976543;
  f) Double $class = 4.5;
  g) int _parents = 20.5;
  h) string name = 'Greg';
```

2. Without doing any programming, what do you think the following main method prints to the screen?

```
void main(List<String> arguments) {
  int x = 5;
  int y = 3;
  int z = x + x * y - y;
  print("The value of z is " + z.toString());

  int w = ++x + y + y--;
  print("The value of w is " + w.toString());
  print("The value of x is now " + x.toString());
  print("The value of y is now " + y.toString());

  bool a = true;
  bool b = false;
  bool c = ((a && (!(x > y))) && (a || y > x));
  print("c is " + c.toString());
}
```

3. Create a new Dart file called `using_operators.dart` and copy the above main method into it. Compile and run. Does the output match what you thought?

4. Create a new Dart file called `temp_converter.dart`. Add a `main` method to `temp_converter.dart` file that declares and initializes a variable to store the temperature in Celsius. Your temperature variable should be store numbers with decimal places.

5. In the `main` method, compute the temperature in Fahrenheit according to the following formula and print it to the screen: `Fahrenheit = (9 ÷ 5) × Celsius + 32`

6. Set the Celsius variable to 100 and compile and run `TempConverter`. The correct output is 212.0. If your output was 132, you probably used integer division somewhere by mistake. [/ vs. ~/].

Dart Lab 2: Control Structures

1. Create a new file called `using_control_structures.dart`.

2. Add a main method to the file, and in the main method declare and initialize a variable to represent a person's age.

3. In the main method, write an `if-else` construct to print out "You are old enough to drive" if the person is old enough to drive and "You are not old enough to drive" if the person is too young.

4. Write a `for` loop that prints out all the odd numbers from 100 to 0 in decreasing order.

5. Do Step 4 with a `while` loop instead of a `for` loop.

Dart Lab 3: Gradebook – Part 1

1. Create a new file called `gradebook.dart`.

2. Add a `main` method of `gradebook.dart`. In the `main` method, declare and initialize a list of `doubles` to store the grades of a student.

3. Write a loop to print out all the grades in the list. Make sure that your printout is readable with spaces or new lines between each grade.

4. Write a new loop to find the sum of all the grades in the list.

5. Divide the sum by the number of grades in the list to find the student's average.

6. Print a message to the user showing the average grade. If the average grade is 85.4, the output should be "Your average grade is 85.4".

7. Your program should work if there are 4 grades in the array or 400 grades in the list. That is, you should be able to change the number of grades in the initialized list and compile, and it should run without any problems. Try it out. If it doesn't, figure out how to rewrite your program so it does.

8. (Optional) Add code to print out the letter grade the student earned based on the average grade. An average in the 90's is an A, in the 80's is a B, 70's is a C, 60's is a D, and anything lower is an F.

Dart Lab 4: Gradebook 2

1. Add a class called `Gradebook` in the `gradebook.dart` file.

1. Add a method to Gradebook called `printGrades` that accepts a list of `doubles` as an argument and prints out all the grades in the list. Replace the loop in the main method that prints out all the grades with a call to the `printGrades` method. Compile and run.

2. Add a method to `Gradebook` called `averageGrade` that takes an array of doubles as an argument and returns the average grade. Replace the loop and calculations in the main method that determines the average grade with a call to the `averageGrade` method. Your main method should still print out the user's average grade and the letter grade the user earned. Compile and run.

3. Change the `main` method of `Gradebook` so that it converts its `String` arguments into `doubles` and initializes the grades in the list to those numbers. Use the method `double.parse` to convert a `String` containing a `double` to an actual `double`. Compile and run and provide arguments at the command line, like this:

```
dart gradebook.dart 82.4 72.5 90 96.8 86.1
```

4. (Optional) Use the `dart.io/stdin` to read from the keyboard. Try modifying your code so that instead of just taking the list of grades from the `main` method, the program asks the user to enter the grades.

5. (Optional) Change the main method so that after asking the user to enter the grades, it prints out a menu of two options for them: 1) print out all the grades or 2) find the average grade. It should ask the user to enter the number of their choice and do what the user chooses.

Dart Lab 5: GradebookOO – Part 1

1. In labs 3 and 4, we built a procedural gradebook program. In labs 5, 6 and 7, we're going to write a new object-oriented gradebook program. Please look back as your `Gradebook` class as needed for help in writing your new object-oriented gradebook.

2. Create a new file called `gradebook_oo.dart` (that's two O's for Object-Oriented) with a class `GradebookOO`. The class should have a single field which is a list of `doubles` called `_grades`.

3. Write a constructor for `GradebookOO`. It takes optional named argument of a list of double and initialize the grades field to an optional value of list of size zero.

4. Add a method to `GradebookOO` named `printGrades` that takes *no* arguments and prints out all the grades in the `_grades` field. Compile.

5. Add a method to `GradebookOO` named `averageGrade` that takes *no* arguments and returns the average grade in the `_grades` field. Compile.

6. Create a new class called `gb_program.dart`. Add a main method to `GBProgram` which instantiates a `Gradebook` with an array of grades, prints out all the grades with a call to the `printGrades` method, and finds the average grade with the `averageGrade` method. Compile and run.

7. (Optional) Use the `dart.io/stdin` in the main method of `GBProgram` to allow the user to enter in the grades.

8. (Optional) Print out a menu to the user, as described in Step 4 of the previous lab, that allows the user to select whether they would like to print out all the grades or find the average grade.

Dart Lab 6: GradebookOO – Part 2

1. Add a method to `GradebookOO` called `addGrade` which accepts a `double` argument and adds it to the list of grades.

2. Delete the `GradebookOO` constructor that takes an array of doubles as an argument. And change the main method of `GBProgram` so that it instantiates an empty `GradebookOO` and adds the grades one-by-one to it with the `addGrade` method. Compile and run.

3. (Optional) If you have not done so in the past few labs, use `dart.io/stdin` to read the grades from the user and print out a menu to the user.

4. (Optional) Add a method `deleteGrade` to `GradebookOO` which accepts a grade as an argument and removes it from the array if it's there. Compile and run.

Dart Lab 7: Racecar – Part 1

1. Create a new file called `racecar.dart` and create a class called `Racecar`. Add two fields to `Racecar`: a field of type String to store the name of the car and a field of type `Color` (from `package:color/color.dart`) to store the color of the car. Each car can have a different name and color.

2. Every one of our racecars will have the same top speed. Add a private constant of type `double` to the `Racecar` class to store the top speed and initialize it to any number you want.

3. Add a constructor to `Racecar` which accepts a name and color argument and assigns the arguments to the name and color fields of the class.

4. Add a getter called `name` that returns the name of the car and a getter called `color` that returns the color of the car.

5. Add a method to `Racecar` called `race` which accepts two `Racecars` as arguments, simulates a race between the two, and returns the car that one the race, or return `null` if the race is a tie. The method should calculate a random speed for each car between 0 and the top speed. The car with the higher speed wins the race, but if they both have the same speed they tie. The method `random` in `java.lang.Math` returns a random double between 0 and 1. If you multiply this random number by the top speed, the product will be a random number between 0 and the top speed. Should this method be static or non-static?.

6. Add a `main` function to `racecar.dart` which creates two `Racecars`, races them against one another, and prints out the winner's name. When instantiating the `Racecar` objects, pass a Color value using the method `Color.rgb()`.

Dart Lab 8: Students – Part 1

1. Create a file called `student .dart` and create a class `Student`, which should have two properties, a name and a year and getters to get the name and get the year of the student. Initialize these properties to arguments passed into the constructor.

3. Create a subclass of `Student` called `Undergrad`. The `Undergrad` constructor should accept name and year arguments. Add a method to `Undergrad` called `description` which returns a `String` containing the name of the undergrad, then a space, then a capital 'U', then a space, and then the year of the undergrad. For example, the `description` method of an `Undergrad` instance with the name "Michael" and the year 2006, should return the `String` `"Michael U 2006"`.

4. Create a subclass of `Student` called `Grad`. The `Grad` constructor should accept only the name of the `Grad` as an argument, and it should always initialize the `Grad`'s year to 5. Add a `description` method to `Grad` which returns a `String` containing the name of the `Grad`, followed by a space and then the letter 'G'. The `description` method of a `Grad` named "Jennifer" should return the `String` `"Jennifer G"`.

5. Create a subclass of `Undergrad` called `Intern`. In addition to the name and year properties, `Intern` should have a wage and a number of hours that are initialized in the constructor. Add a `getPay` method to `Intern` which returns the wage times the number of hours. Add a `description` method to `Intern` which returns a `String` containing the result of calling `Undergrad`'s description method followed by the return value of the `getPay` method. The `description` method of an `Undergrad` named "Elizabeth" whose year is 2005 and worked 20 hours at $10.32/hour, should return the `String` `"Elizabeth U 2005 206.4"`.

6. Create a subclass of `Grad` called `ResearchAssistant`. `ResearchAssistant` has a salary that is initialized in the constructor and a `getPay` method that returns the salary. Add a `description` method to `ResearchAssistant` which returns a `String` containing the result of `Grad`'s `description` method, followed by the result of `getPay`. The `description` method of a `ResearchAssistant` with the name "Greg" and a $2000.00 salary would return the `String` `"Greg G 2000.0"`.

7. Create a file called `student_test.dart` that has a main method. Use the main method to test the class hierarchy you just built. Create some instances of `Undergrad`, `Grad`, `Intern`, and `Research Assistant`. Print out the result of their `description` methods. Compile and run.

Dart Lab 9: Students – Part 2

1. A university tells you they want to use the student objects that you built in their new software system. For the rest of the lab, you will be improving your student objects to meet the needs of the university.

2. The university wants to be able to easily print out descriptions of every student. In the main method of `student_test.dart`, add the instances of `Undergrad`, `Grad`, `Intern`, and `Research Assistant` that you created in the last lab to an `List` of `Student`. Iterate through the `List`, cast each element to a `Student` and print out the return value of the `description` method of each. Try to compile. The call to the description method should generate a "cannot resolve symbol" error. Why?

3. Fix the error by adding a dummy `description` method to `Student` which returns `''`. Compile and run.

4. The university tells you that they do not want the `Student` class to be instantiated, and they want to guarantee that every subclass of `Student` implements the `description` method. Change the `Student` class so it meets these two requirements. Compile and run.

5. The university wants to be able to use your objects in their student payroll system. They need to be able to easily print out the pay of all the interns and research assistants. In the main method of

student_test.dart, create an List with just Interns and ResearchAssistants. Is it possible to iterate through the List, cast each to a Student, and call getPay on each?

5. Create an Employee interface with a single method, getPay. Have Intern and ResearchAssistant implement that interface. Now iterate through your list of employees and print out the pay of each.

Dart Lab 10: MyStore – Part 1

1. In the next two labs you will write software to run a store. The store can sell any products you want – it's up to you.

2. Create a new file called products.dart and create a class called Product. This will represent a product sold in your store. Add fields to the Product class to store the name and price of the product. These fields should be assigned to values passed as arguments into the constructor. Add getters to Product to return the name and price of the product. Override the toString() method to print both the name and price of a Product. Compile.

3. Change the Product constructor so it throws a NullThrownError if the name is null and an ArgumentError if the price is negative. Compile.

4. Create a new file called my_store.dart with a class MyStore. my_store.dart should have a List field to contain all the products in your store. Initialize the field to an empty list. Compile.

5. Add a method to MyStore called readProducts. This method should accept no arguments and return nothing. In the readProducts method, print messages to the console that ask the user to enter in a product name and price, and then read the name and price with dart.io/stdin. Instantiate a Product with that name and price and add that product to the list of products. Compile.

6. Add a main method to my_store. In the main method, instantiate a MyStore object and call the readProducts method on that object. Compile and run.

7. What happens when you type a word instead of a number when your program asks you to type in a price?

8. Write a new custom exception called ProductException inside a file my_exceptions.dart. Make sure it implements the Exception interface. Compile.

9. Change the readProducts method in MyStore so that it catches the FormatException thrown by stdin and throws a ProductException inside the catch clause. Change the main

method in MyStore so that it catches a ProductException and prints out an appropriate message to the user. Compile and run.

Dart Lab 11: MyStore – Part 2

1. Using Notepad, start a new text file and save it as products.txt to the same directory of the my_store.dart file. On each line of the file, write the name of a product you want your store to sell, a '#' symbol, and then the price of the product. For example, if you want your store to sell a computer for 1500.45 and a soccer ball for 37.23, make sure your products.txt file has the following two lines:

```
Computer#1500.45
soccer ball#37.23
```

Your products.txt file should list at least 10 products.

2. Add a method to MyStore called readProductsFromFile. The method should accept one argument, a String containing a filename. Open the file with that filename using a File().readAsStringSync() to read the file line-by-line. For each line of the file, split the string line at '\n' character then using a loop split each line at '#' character. Use the return list from the split() method to get name and price of the product. Then convert the price from a String to a number and instantiate a Product object with that name and price. Finally, add that product to the store's list of products. Catch any Exception thrown by the above operations and throw a ProductException in the catch clause. Compile.

3. In the main method of my_store.dart, call the readProductsFromFile method on the store instance you created and pass the String "products.txt" as an argument to the method. Compile and run.

4. (Optional) Write an interface called ProductSource inside a file product_source.dart which has a single method, getProducts that accepts no arguments and returns a list of products. Write two classes that implement ProductSource. The first, called KeyboardSource, should have an empty constructor. Its getProducts method should ask the user to type in some products the same way your readProducts method does, and it should return a list of products that the user typed in. The second class, FileSource, should accept a filename argument, and its getProducts method should read products from the file like the readProductsFromFile method and return a list of products in the file. Compile.

7. (Optional) Replace the readProducts and readProductsFromFile method with a single method loadProducts method that accepts a ProductSource argument. The loadProducts

method should call `getProducts` on the `ProductSource` and add the products returned by the `ProductSource` to the store's list of products. Hint: using the `addAll` method provided by lists should make your life easier. Compile.

8. (Optional) Change the main method of `my_store.dart` to use `KeyboardSource`, `FileSource`, and the `loadProducts` method instead of the `readProducts` and `readProductsFromFile` methods. Why is this a better design?