LAB-4

AIM : Collections and Advanced Classes

Prac-1 : Create a map with the following keys: name, profession, country and city. Forthe values, add your own information.

Code :

```
void main() {
 Map<String, String> myMap = {
   'Name' : "Vineet Joshi",
   'Profession' : "Student",
   'Country' : "India",
   'City' : "Nadiad"
 };

  print(myMap);
}
```

Output :

```
[Running] dart "f:\B.
Name = Vineet Joshi
Profession = Student
Country = India
City = Nadiad
```

Prac-2 : You suddenly decide to move to Toronto, Canada. Programmatically update the values for country and city.

Code :

```
void main() {
 Map<String, String> myMap = {
   'Name' : "Vineet Joshi",
   'Profession' : "Student",
   'Country' : "India",
   'City' : "Nadiad"
 };
```

```
  print("Before : $myMap");

  myMap['Country'] = 'Canada';
  myMap['City'] = 'Toronto';

  print("After : $myMap");
}
```

Output :

```
[Running] dart "f:\B.TECH\SEM-5\SDP\lab_4\tempCodeRunnerFile.dart"
Before : {Name: Vineet Joshi, Profession: Student, Country: India, City: Nadiad}
After : {Name: Vineet Joshi, Profession: Student, Country: Canada, City: Toronto}
```

Prac-3 : Iterate over the map and print all the values.

Code :

```
void main() {
 Map<String, String> myMap = {
   'Name' : "Vineet Joshi",
   'Profession' : "Student",
   'Country' : "India",
   'City' : "Nadiad"
 };

  myMap.forEach((key, value) => print("$key = $value"));
}
```

Output :

```
[Running] dart "f:\B.T
Name = Vineet Joshi
Profession = Student
Country = India
City = Nadiad
```

Prac-4 : Write a function that takes a paragraph of text and returns a collection of unique String characters that the text contains.

Code :

```
Set<String> uniqueCharactersInParagraph(String paragraph) {
  paragraph = paragraph.toLowerCase();
  Set<String> uniqueChars = Set();

  for (var char in paragraph.runes) {
    String charString = String.fromCharCode(char);

    if (charString.replaceAll(RegExp(r'[^a-zA-Z]'), '').isNotEmpty) {
      uniqueChars.add(charString);
    }
  }

  return uniqueChars;
}

void main() {
  String paragraph = "The Moon is a barren, rocky world without air and water. It has dark
lava plain on its surface. The Moon is filled wit craters. It has no light of its own.";
  Set<String> uniqueChars = uniqueCharactersInParagraph(paragraph);
  print(uniqueChars);
}
```

Output :

```
[Running] dart "f:\B.TECH\SEM-5\SDP\lab_4\tempCodeRunnerFile.dart"
{t, h, e, m, o, n, i, s, a, b, r, c, k, y, w, l, d, u, v, p, f, g}
```

Prac-5 : Repeat Challenge 1, but this time have the function return a collection that contains the frequency, or count, of every unique character.

Code :

```
Map<String, int> calculate(String paragraph) {
  var words = paragraph.split(" ");
  Map<String, int> result = {};

  for(var word in words) {
    if(result[word] == null) {
      result[word] = 1;
    } else {
      result[word] = result[word]! + 1;
    }
```

```
  }
  return result;
}
void main() {
  var paragraph = "The Moon is a barren, rocky world without air and water. It has dark lava plain on its surface. The Moon is filled wit craters. It has no light of its own.";
  Map<String, int> result = calculate(paragraph);
  print(result);
}
```

Output :

```
[Running] dart "f:\B.TECH\SEM-5\SDP\lab_4\tempCodeRunnerFile.dart"
{The: 2, Moon: 2, is: 2, a: 1, barren,: 1, rocky: 1, world: 1, without: 1, air: 1, and: 1, water.: 1,
It: 2, has: 2, dark: 1, lava: 1, plain: 1, on: 1, its: 2, surface.: 1, filled: 1, wit: 1, craters.: 1,
no: 1, light: 1, of: 1, own.: 1}
```

Prac-6 : Create a class called User with properties for id and name. Make a List with three users, specifying any appropriate names and IDs you like. Then write a function that converts your user list to a list of maps whose keys are id and name.

Code:

```
class user {
  int id = 0;
  String name = "";

  user(int id, String name ) {
    this.id = id;
    this.name = name;
  }
}

Map<int, String> convert(var users) {
  Map<int, String> ans = {};

  for(var user in users) {
    ans[user.id] = user.name;
  }
```

```
  return ans;
}
void main() {
  user u1 = user(1, "Micheal");
  user u2 = user(2, "Oliver");
  user u3 = user(3, "Prometheus");

  var users = [u1, u2, u3];

  Map<int, String> converted = convert(users);
  print(converted);
}
```

Output:

```
[Running] dart "f:\B.TECH\SEM-5\SDP\lab_4\
{1: Micheal, 2: Oliver, 3: Prometheus}
```

Prac-7 : Create a class named Fruit with a String field named color and a method named describeColor, which uses colorto print a message.
Create a subclass of Fruit named Melon and then create two Melon subclasses named Watermelon and Cantaloupe.
Override describeColorin the Watermelon class to vary the output.

Code:

```
class Fruit {
  String color = "";

  String describeColor() {
    return "Color is $color";
  }
}
```

```
class melon extends Fruit {}

class Watermelon extends melon {
  @override
  String describeColor() {
    return "Color of Watermelon is ${this.color}";
  }
}

class Cantaloupe extends Fruit {}

void main() {
  final f1 = Watermelon();
  final f2 = Cantaloupe();
  f1.color = "red";
  f2.color = "black";
  print(f1.describeColor());
  print(f2.describeColor());
}
```

Output:

```
[Running] dart "f:\B.TECH\SE
Color of Watermelon is red
Color is black
```

Prac-8 : Create an interface called Bottle and add a method to it called open.
Create a concrete class called SodaBottle that implements Bottle and prints “Fizz fizz” when open is called.
Add a factory constructor to Bottle that returns a SodaBottle instance.
Instantiate SodaBottle by using the Bottle factory constructor and call open on the object.

Code:

```
abstract class Bottle {
  factory Bottle() => sodaBottle();
  void open();
}

class sodaBottle implements Bottle {
  @override
  void open() {
    print("Fizz Fizz");
  }
}

void main() {
  final obj = Bottle();
  obj.open();
}
```

Output:

```
[Running] da
Fizz Fizz
```

Prac-9 : Create a class called calculator with a method called sum that prints the sum of any two integers you give it. Extract the logic in sum to mixin called adder. use the mixin in calculator

Code:

```
mixin Adder {
  num sub(a, b) {
    return a - b;
  }

  num mul(a, b) {
    return a * b;
```

```
  }
}

class Calculator {
  num sum(a, b) {
    return a + b;
  }
}

class calc extends Calculator with Adder {

}

void main() {
  final c = calc();
  print(c.sum(2, 3));
  print(c.sub(2, 3));
  print(c.mul(2, 3));
}
```

Output:

```
[Running]
5
-1
6
```

Prac-10 : Dart has a class named Comparable, which is used by the the sort method of List to sort its elements. Add a weight field to the Platypus class you made in this lesson. Then make Platypus implement Comparable so that when you have a list of Platypus objects, calling sort on the list will sort them by weight.Override describeColorin the Watermelon class to vary the output.

Code:

```
class Platypus implements Comparable<Platypus> {
  String name;
  double weight;

  Platypus(this.name, this.weight);

  @override
  int compareTo(Platypus other) {
    return weight.compareTo(other.weight);
  }

  @override
  String toString() {
    return name;
  }
}

void main() {
  List<Platypus> list = [
      Platypus("Perry", 2.5),
      Platypus("Patsy", 1.8),
      Platypus("Peter", 3.2),
    ];

  list.sort();
  print(list);
}
```

Output:

```
[Running] dart "f:\B.TEC
[Patsy, Perry, Peter]
```

Prac-11 : Dart has a Duration class for expressing lengths of time.
Make an extension on int so that you can express a duration
like so:

Code:

```
extension DurationExtension on int {
  Duration get seconds => Duration(seconds: this);
  Duration get minutes => Duration(minutes: this);
  Duration get hours => Duration(hours: this);
  Duration get days => Duration(days: this);
}

void main() {
  final timeRemaining = 3, minutes;
  print(timeRemaining.toString());
}
```

Output:

```
[Running
3
```