IEEE FCI - Helwan Student Branch

S25' Training Sessions

[illegible]

SESSION 3

#create_share_innovate

IEEE | FCI - Helwan Student Branch

[illegible]

Introduction to Object-Oriented Programming (OOP) → Collections → Operations on collections → Collection Methods

S25' Training Sessions

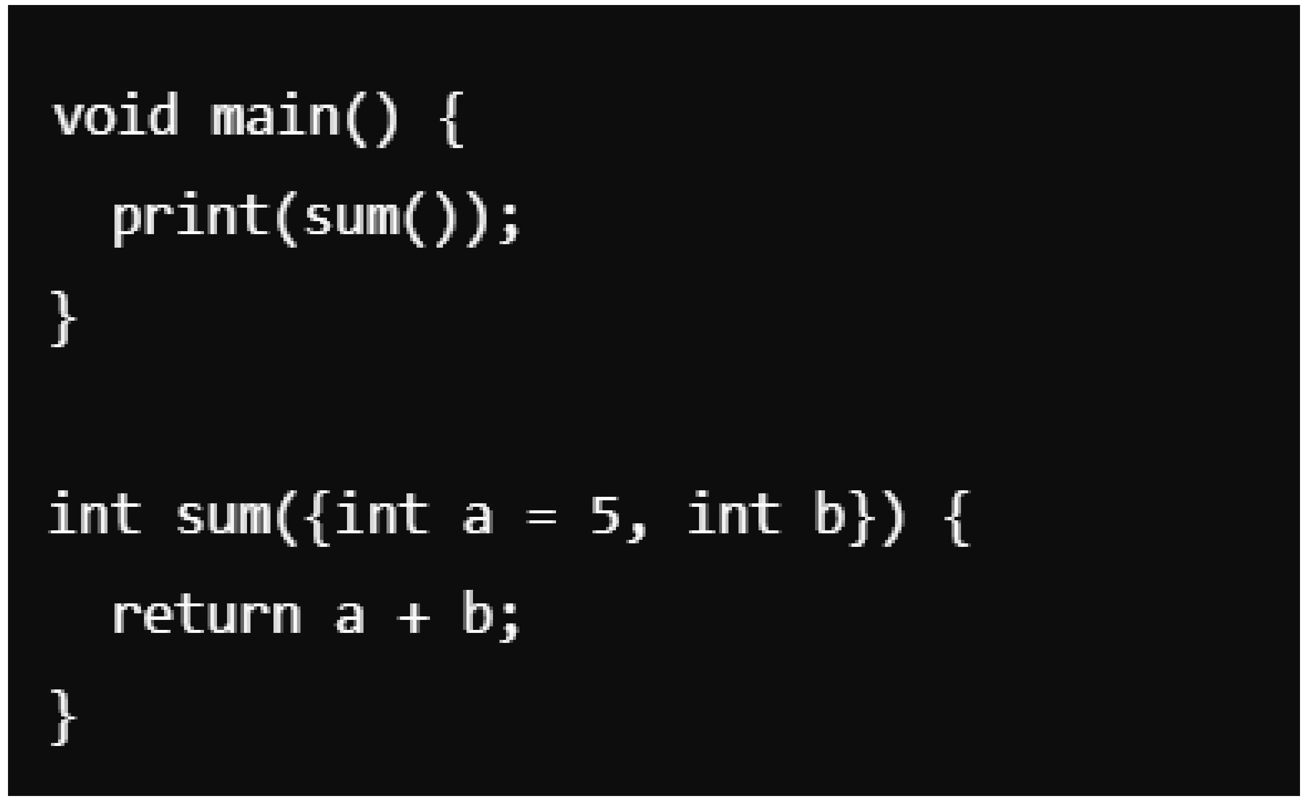
IEEE | FCI - Helwan Student Branch

[illegible]

RECAP

uiz

What will happen if you run this code?



```
void main() {
  print(sum());
}

int sum({int a = 5, int b}) {
  return a + b;
}
```

uiz

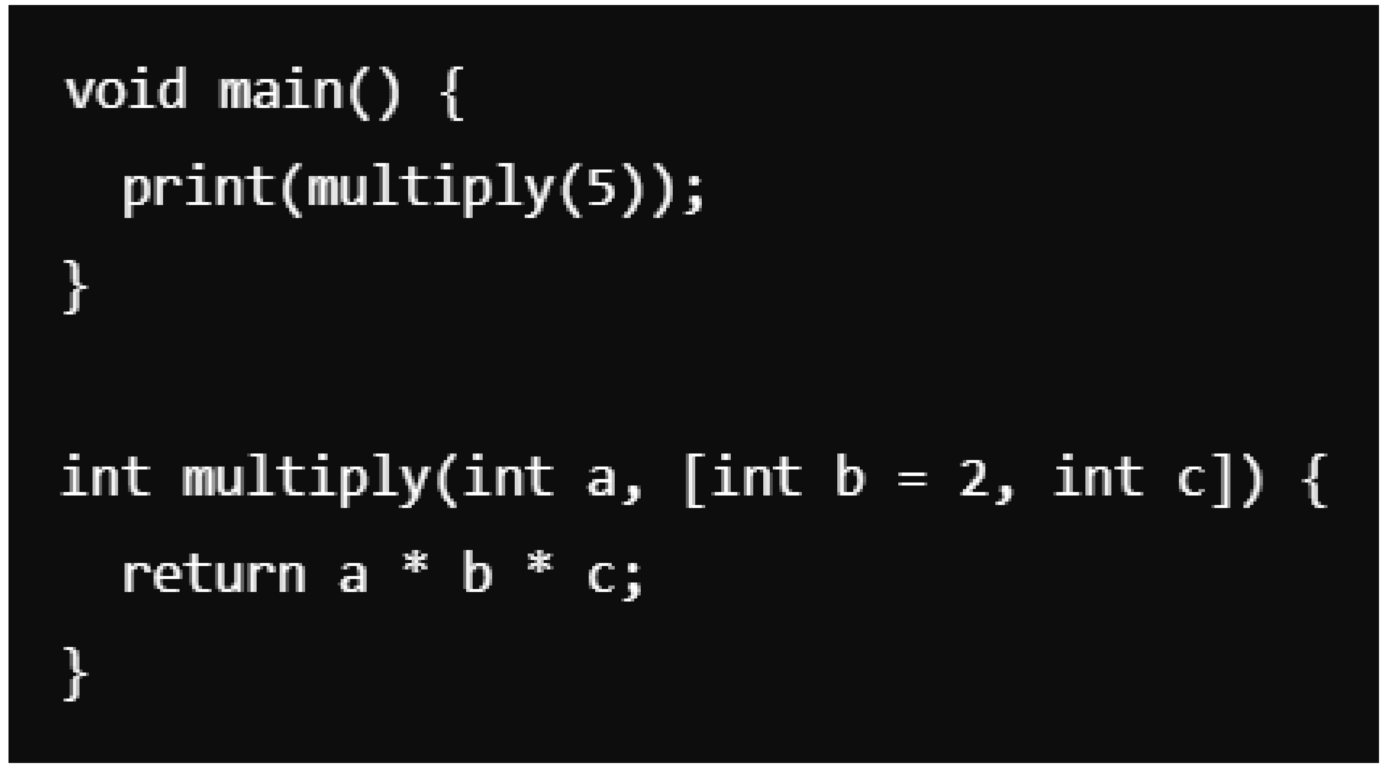
Answer: Compilation error

IEEE FCI - Helwan Student Branch

UIZ

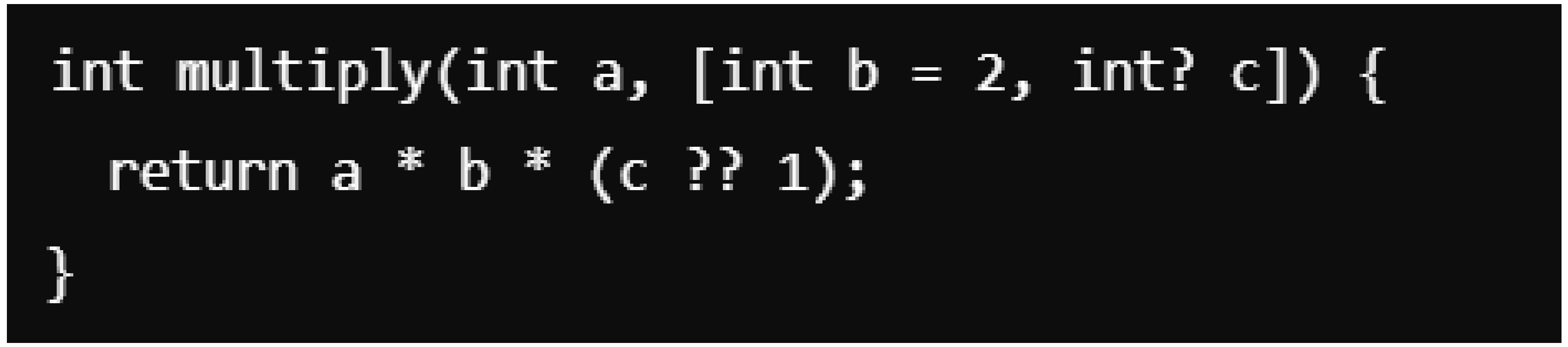
S25' Training Sessions

Issue in this function and fix it?



```
void main() {
  print(multiply(5));
}

int multiply(int a, [int b = 2, int c]) {
  return a * b * c;
}
```

Answer: Provide a default value: [int c = 1],
or use a nullable type and null-aware operator:

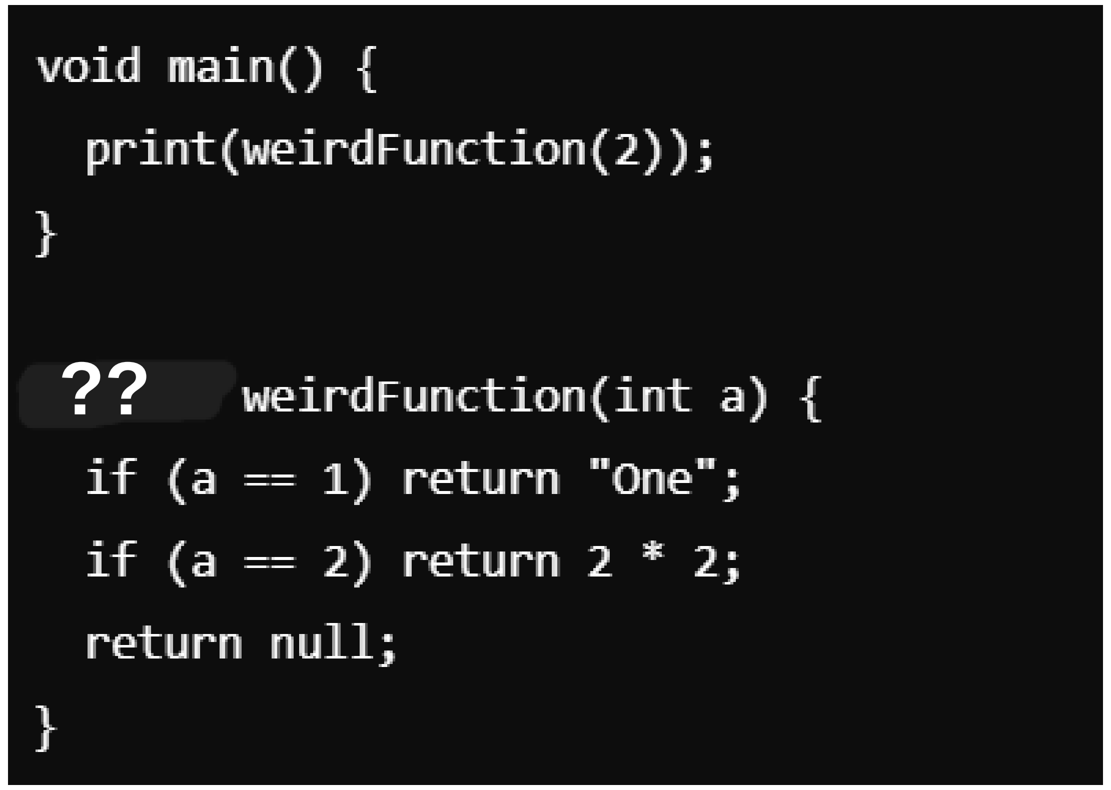


```
int multiply(int a, [int b = 2, int? c]) {
  return a * b * (c ?? 1);
}
```

IEEE | FCI - Helwan Student Branch

UIZ

Return type of weirdFunction?



```
void main() {
  print(weirdFunction(2));
}

??    weirdFunction(int a) {
  if (a == 1) return "One";
  if (a == 2) return 2 * 2;
  return null;
}
```

S25' Training Sessions

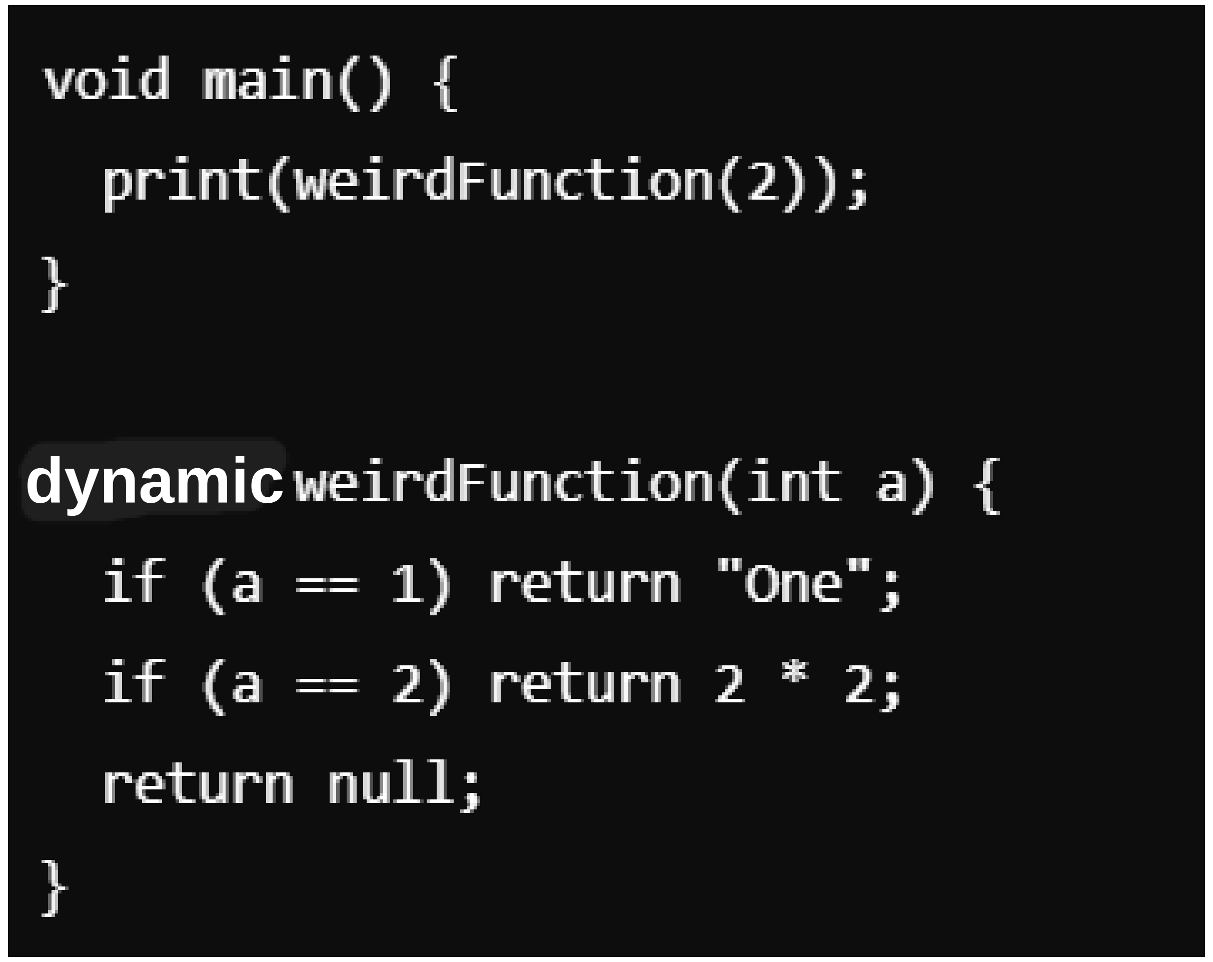
IEEE | FCI - Helwan Student Branch

S25' Training Sessions

Answer:

dynamic

dynamic



```
void main() {
  print(weirdFunction(2));
}

dynamic weirdFunction(int a) {
  if (a == 1) return "One";
  if (a == 2) return 2 * 2;
  return null;
}
```

What does the following ternary operator do?

condition ? trueValue : falseValue

A) Executes the trueValue if the condition is true, otherwise executes falseValue

B) Always executes trueValue

C) Always executes falseValue

D) Only works inside a switch statement

IEEE | FCI - Helwan Student Branch

UIZ

??

Answer:

A) Executes the trueValue if the condition is true, otherwise executes falseValue

S25' Training Sessions

IEEE | FCI - Helwan Student Branch

[illegible]

A programming paradigm based on objects.

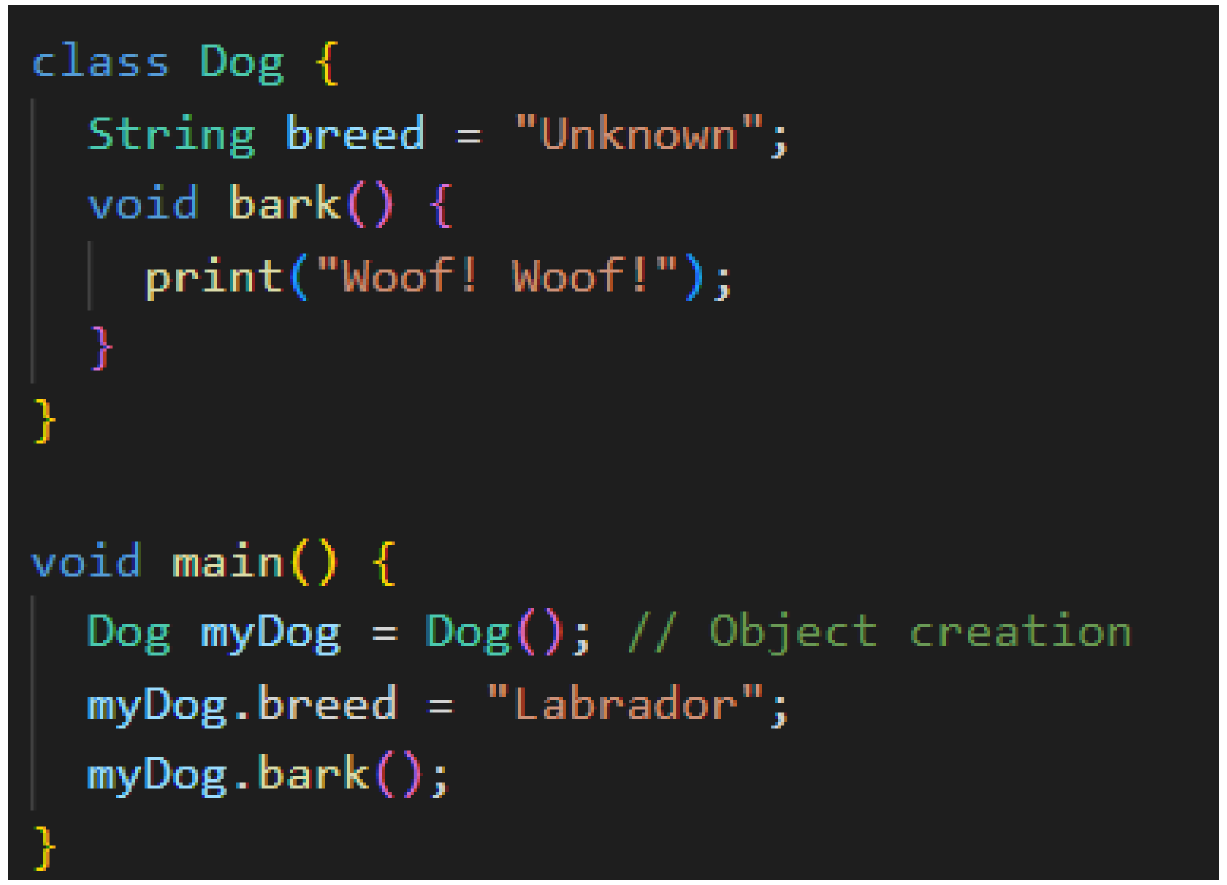
- Uses real-world modeling for better code organization.
- Focuses on **data** and **behavior** together.
- **concepts**: Encapsulation, Abstraction, Inheritance, Polymorphism.

IEEE FCI - Helwan Student Branch

[illegible]

What are Classes and Objects?

Class: Blueprint for creating objects.
Object: An instance of a class.
Objects have **state (attributes)** and **behavior (methods)**.



```
class Dog {
  String breed = "Unknown";
  void bark() {
    print("Woof! Woof!");
  }
}

void main() {
  Dog myDog = Dog(); // Object creation
  myDog.breed = "Labrador";
  myDog.bark();
}
```

S25' Training Sessions

IEEE FCI - Helwan Student Branch

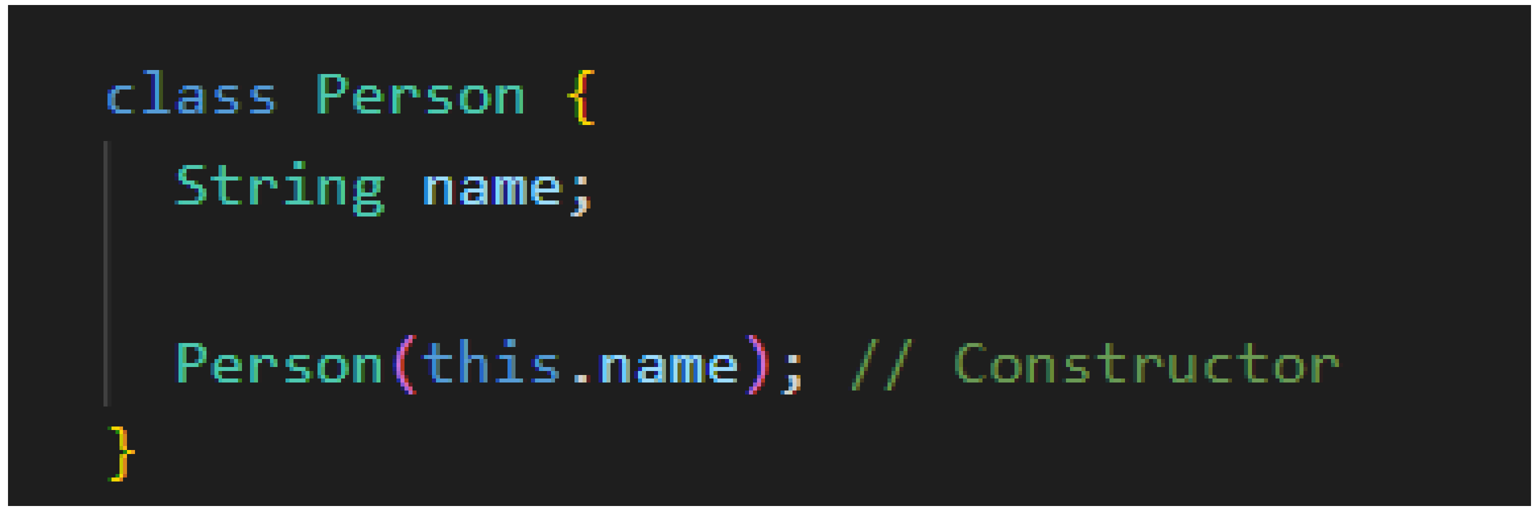
CONSTRUCTORS

What are constructors

Special method used to initialize objects.

Called automatically when an object is created.

Name is the same as the class.



```
class Person {
  String name;

  Person(this.name); // Constructor
}
```

IEEE FCI - Helwan Student Branch

[illegible]

Types of constructors:

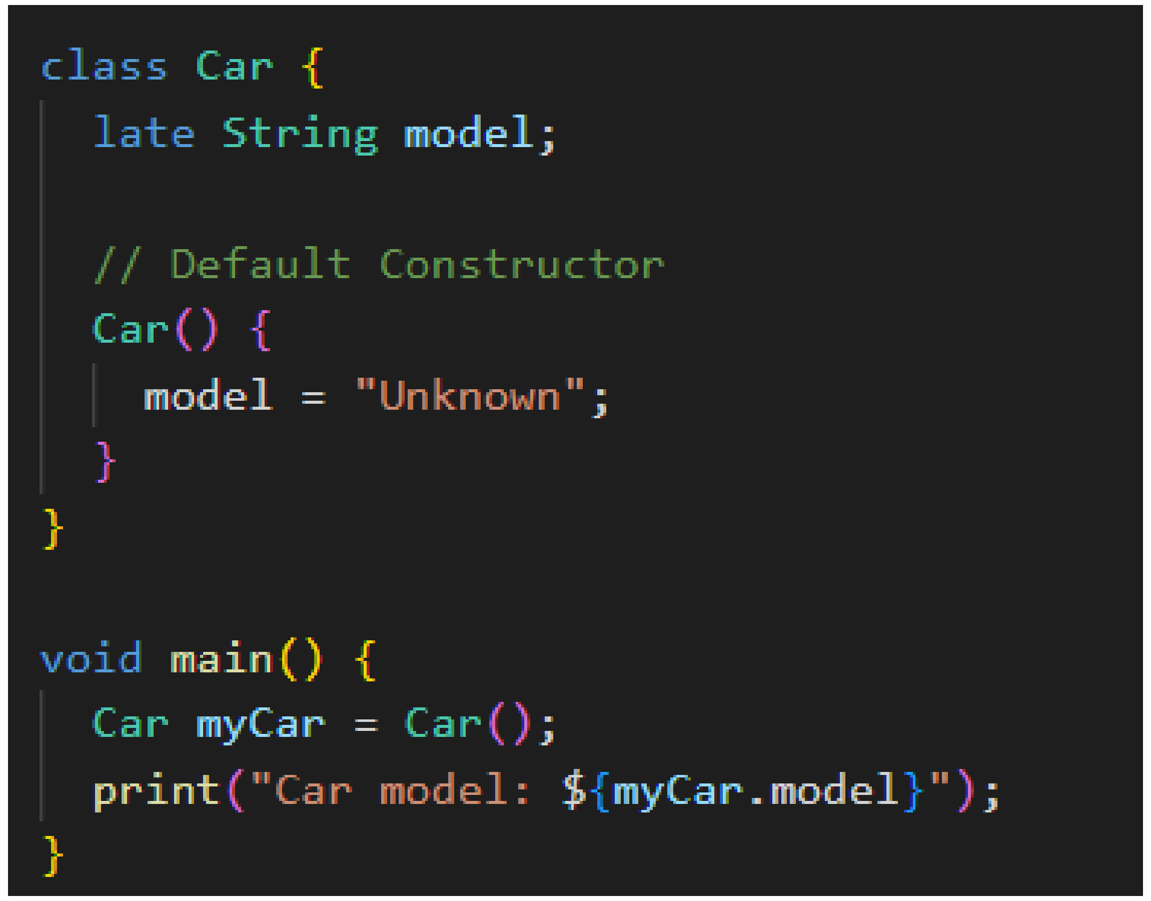
1. Default Constructor (No parameters)
2. Parameterized Constructor (Accepts parameters)
3. Named Constructor.

IEEE | FCI - Helwan Student Branch

[illegible]

Default Constructor (No parameters)

- A constructor that takes no arguments.
- Initializes objects with default values.
- If no constructor is defined, Dart provides a default .
- constructor automatically



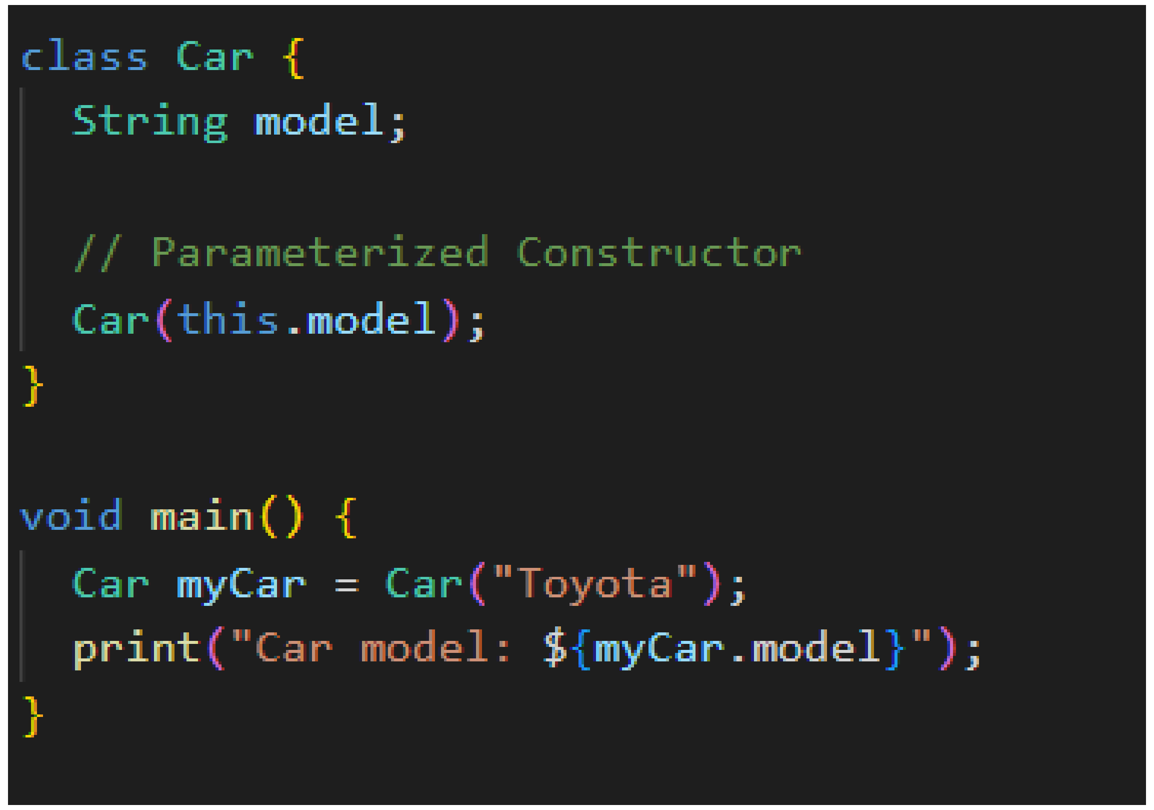
```
class Car {
  late String model;

  // Default Constructor
  Car() {
    model = "Unknown";
  }
}

void main() {
  Car myCar = Car();
  print("Car model: ${myCar.model}");
}
```

S25' Training Sessions

[illegible]

Parameterized Constructor (Accepts parameters)

- A constructor that takes arguments to initialize attributes.
- Useful for assigning custom values during object creation.

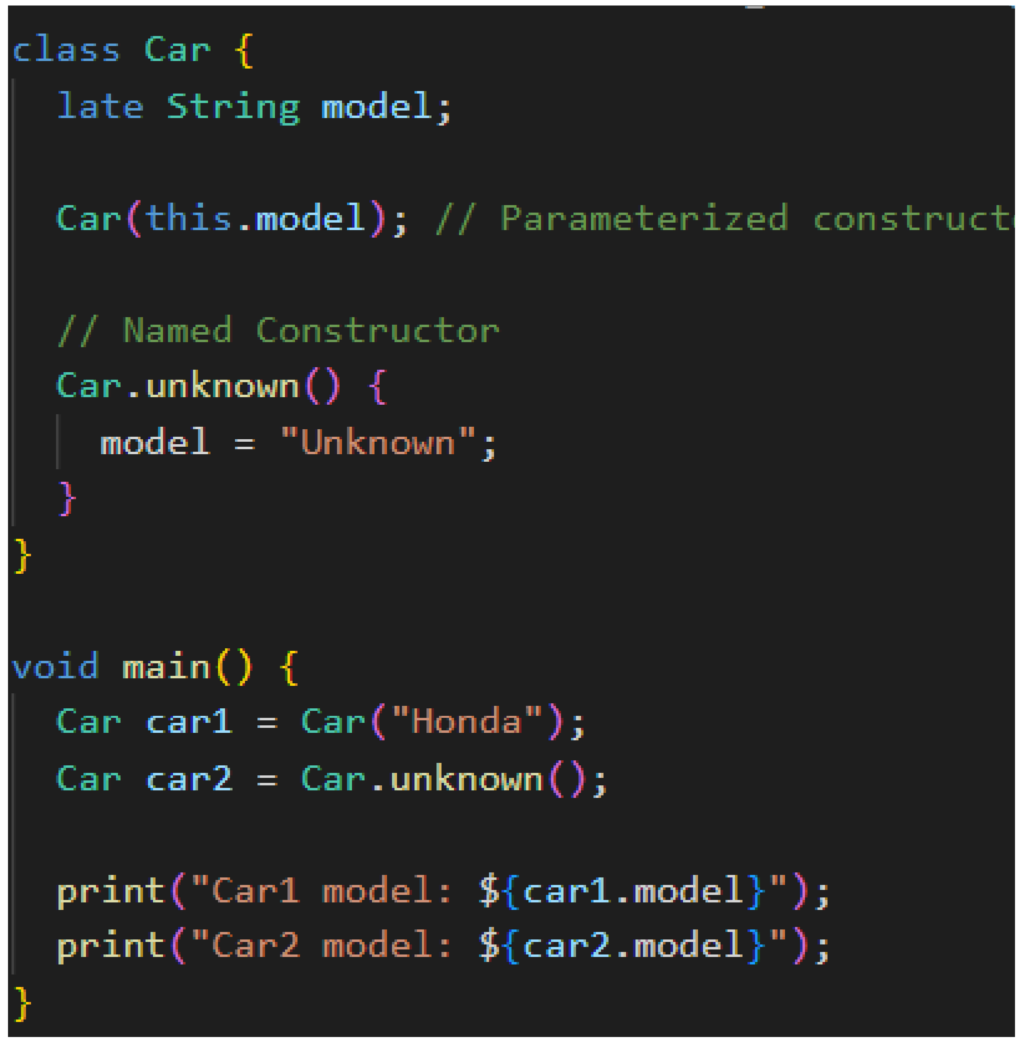


```
class Car {
  String model;

  // Parameterized Constructor
  Car(this.model);
}

void main() {
  Car myCar = Car("Toyota");
  print("Car model: ${myCar.model}");
}
```

[illegible]

Named Constructor

(Provides additional ways to create objects)



```
class Car {
  late String model;

  Car(this.model); // Parameterized construct

  // Named Constructor
  Car.unknown() {
    model = "Unknown";
  }
}

void main() {
  Car car1 = Car("Honda");
  Car car2 = Car.unknown();

  print("Car1 model: ${car1.model}");
  print("Car2 model: ${car2.model}");
}
```

IEEE | FCI - Helwan Student Branch

EXERCISE

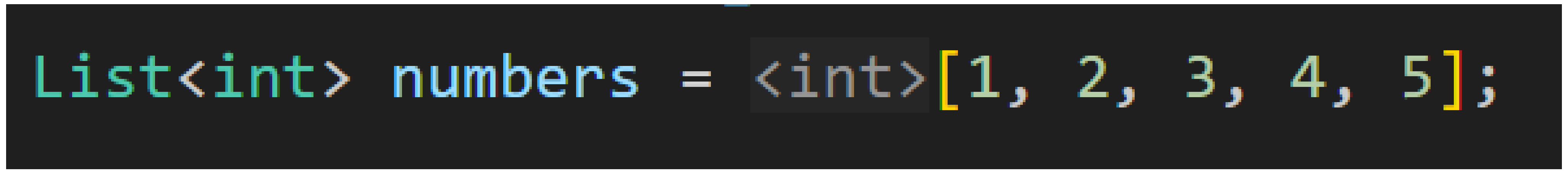
Define a class Book with attributes title and author.

- Create a constructor to initialize these attributes.
- Instantiate an object using the constructor.

[illegible]

List:

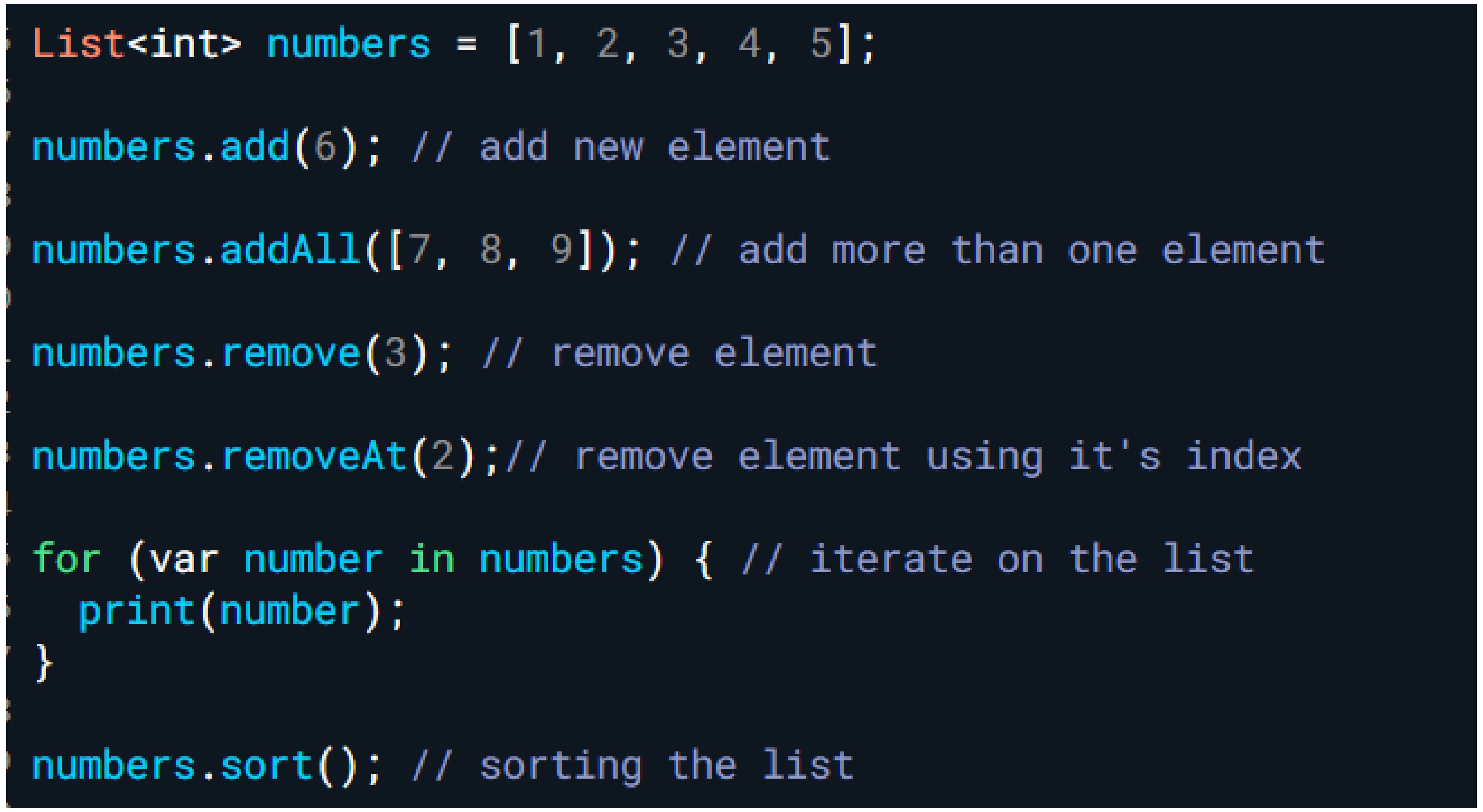
A list is an ordered collection of items.



```
List<int> numbers = <int>[1, 2, 3, 4, 5];
```

[illegible]

List operations:



```dart
List<int> numbers = [1, 2, 3, 4, 5];

numbers.add(6); // add new element

numbers.addAll([7, 8, 9]); // add more than one element

numbers.remove(3); // remove element

numbers.removeAt(2);// remove element using it's index

for (var number in numbers) { // iterate on the list
  print(number);
}

numbers.sort(); // sorting the list
```

IEEE | FCI - Helwan Student Branch

[illegible]

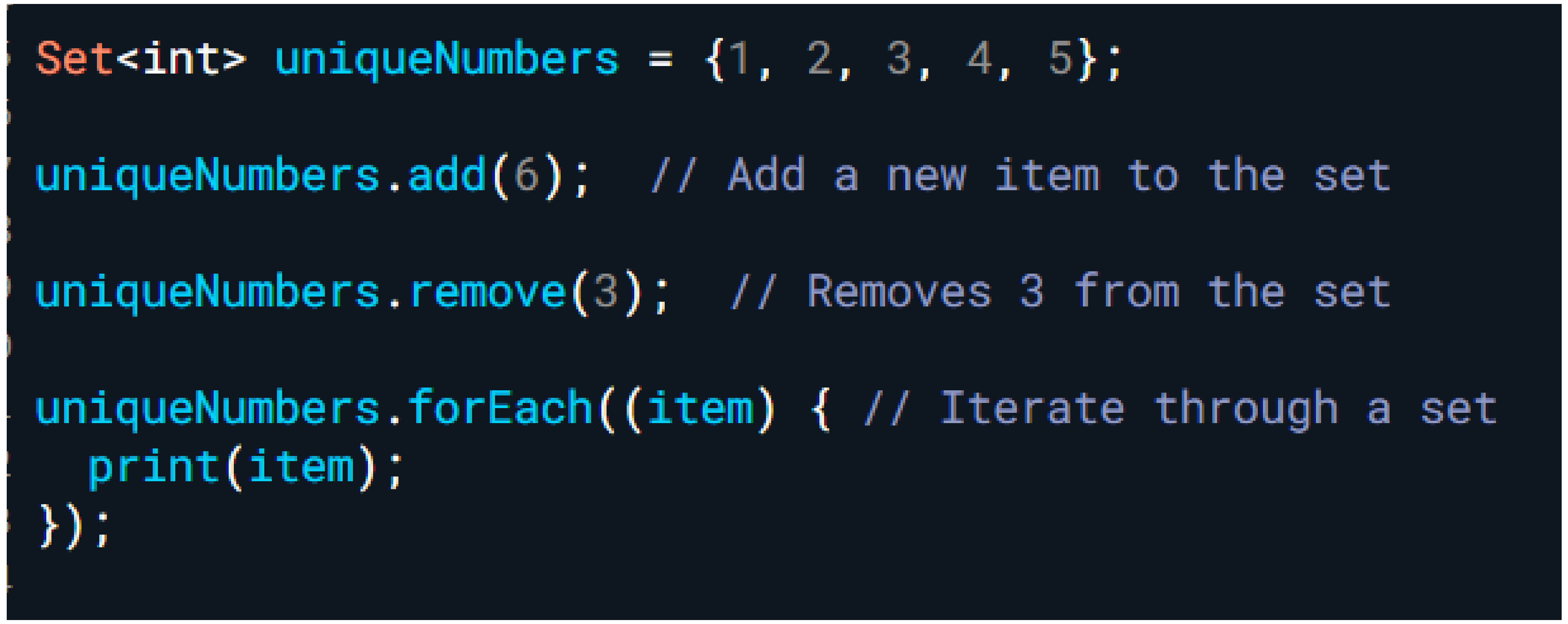
Set:

A set is an unordered collection of unique items.

```
Set<int> uniqueNumbers = <int>{1, 2, 3, 4, 5};
```

[illegible]

Set operations:



```
Set<int> uniqueNumbers = {1, 2, 3, 4, 5};

uniqueNumbers.add(6);  // Add a new item to the set

uniqueNumbers.remove(3);  // Removes 3 from the set

uniqueNumbers.forEach((item) { // Iterate through a set
  print(item);
});
```

FCI - Helwan
Student Branch

[illegible]

Map:

A map is a collection of key-value pairs.

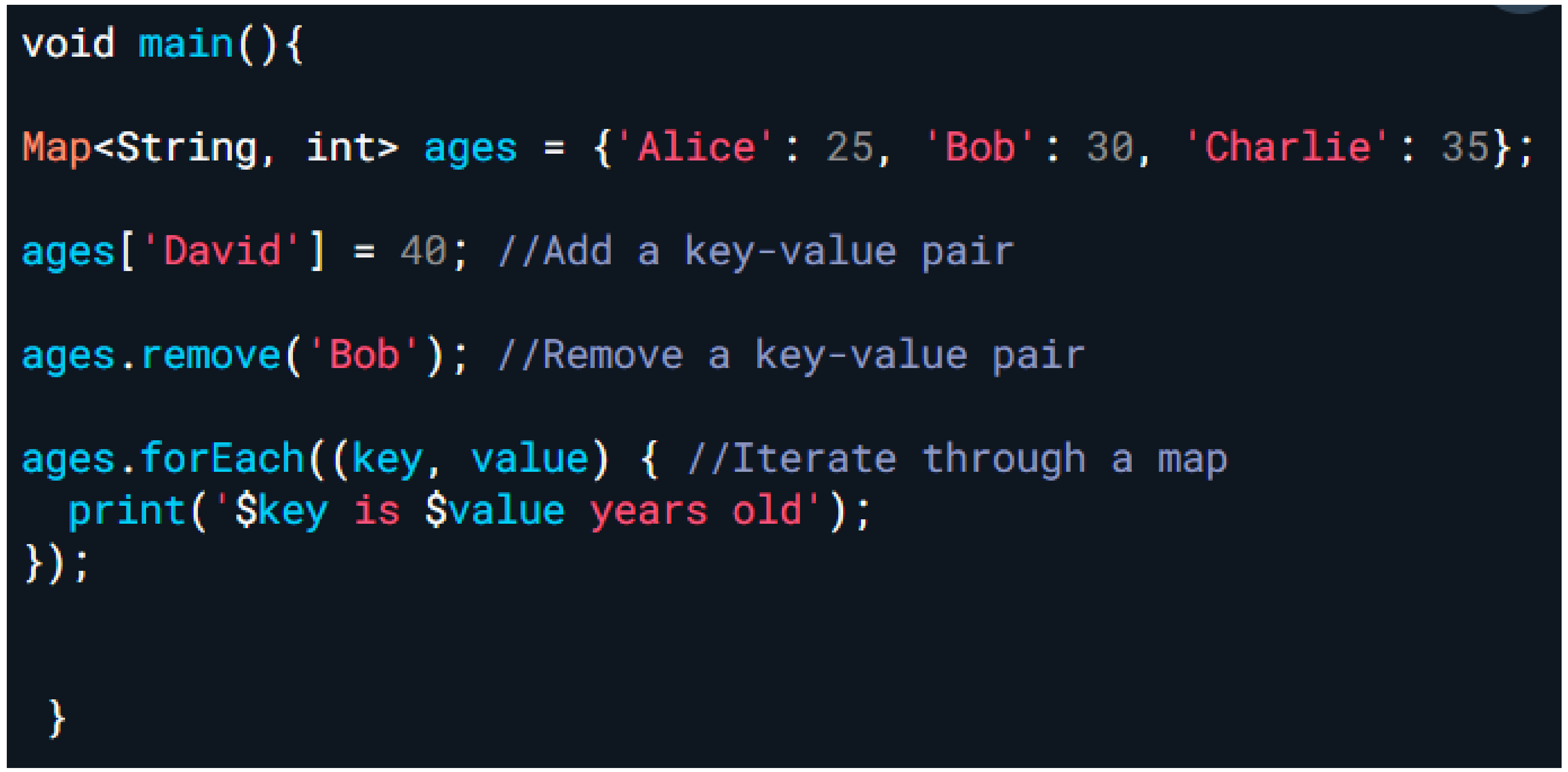
```
Map<String, int> ages = <String, int>{'Alice': 25, 'Bob': 30, 'Charlie': 35};
```

S25' Training Sessions

IEEE | FCI - Helwan Student Branch

COLLECTIONS

Map operations:



```
void main(){

Map<String, int> ages = {'Alice': 25, 'Bob': 30, 'Charlie': 35};

ages['David'] = 40; //Add a key-value pair

ages.remove('Bob'); //Remove a key-value pair

ages.forEach((key, value) { //Iterate through a map
  print('$key is $value years old');
});


 }
```

IEEE | FCI - Helwan Student Branch

[illegible]

Collection Methods

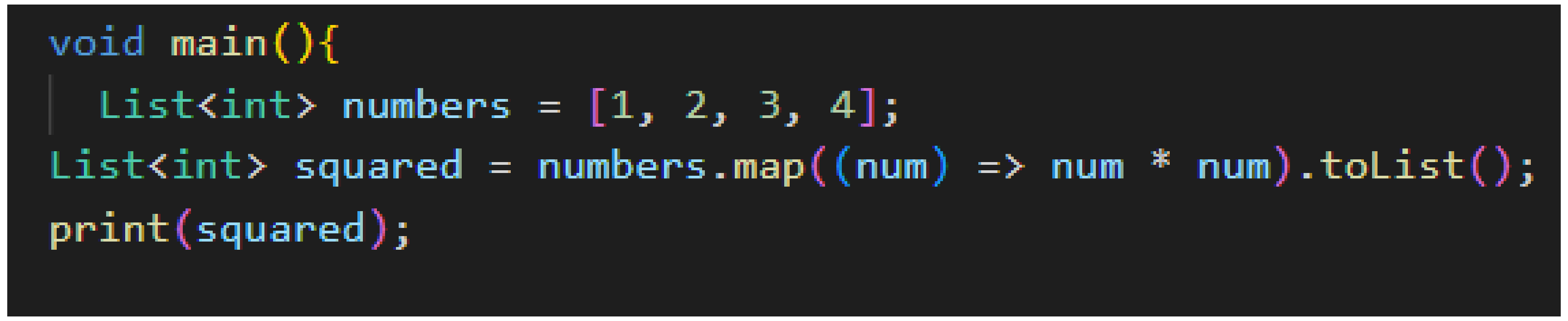
:

In Dart, collections (such as List, Set, and Map) come with higher-order functions that accept anonymous functions as parameters. These functions allow you to transform, filter, iterate, or reduce collections without writing explicit loops.

[illegible]

1.map() (Transform Elements)

The map() function applies a transformation to each element of the collection and returns a new iterable.
It does not modify the original list.
You must use **.toList()** or .toSet() if you need a concrete collection.



```
void main(){
  List<int> numbers = [1, 2, 3, 4];
List<int> squared = numbers.map((num) => num * num).toList();
print(squared);
```

[illegible]

2. forEach() (Iterating Over Elements)

What It Does:

Executes a function for each element in the collection.

Unlike map(), it does not return a new collection.

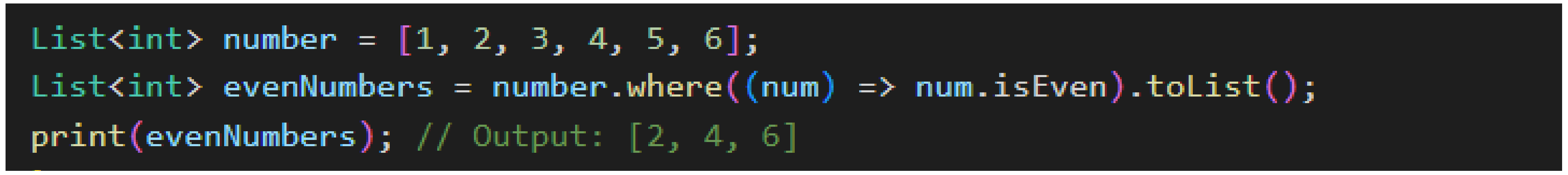
```
List<String> name = ["Alice", "Bob", "Charlie"];
name.forEach((name) => print("Hello, $name!"));
```

[illegible]

3. where() (Filtering Elements)

What It Does:

- Returns a new collection containing only the elements that match a condition.
- Does not modify the original collection.
- You must use **.toList()** or **.toSet()** if you need a concrete collection.



```
List<int> number = [1, 2, 3, 4, 5, 6];
List<int> evenNumbers = number.where((num) => num.isEven).toList();
print(evenNumbers); // Output: [2, 4, 6]
```

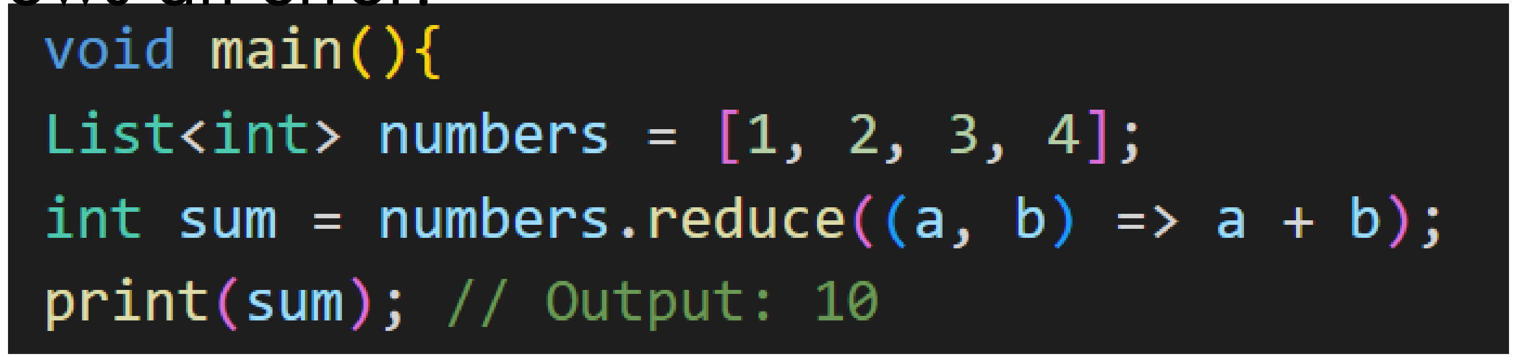
IEEE FCI - Helwan Student Branch

[illegible]

4.reduce() (Combining Elements into One Value)

What It Does:

- Iterates over the collection and combines elements into a single result.
- The function must take two arguments (previous result and current element).
- If the collection is empty, it throws an error.



```dart
void main(){
List<int> numbers = [1, 2, 3, 4];
int sum = numbers.reduce((a, b) => a + b);
print(sum); // Output: 10
```

S25' Training Sessions

IEEE | FCI - Helwan Student Branch

EXERCISE

Build a program to manage a supermarket inventory using a Map for items and their prices. Include functions to add, remove, and update items.

FCI - Helwan
Student Branch

LINK

Refactor the supermarket program to include user input for dynamic inventory management

IEEE | FCI - Helwan Student Branch

THANK YOU

SEE YOU NEXT TIME

S25' Training Sessions