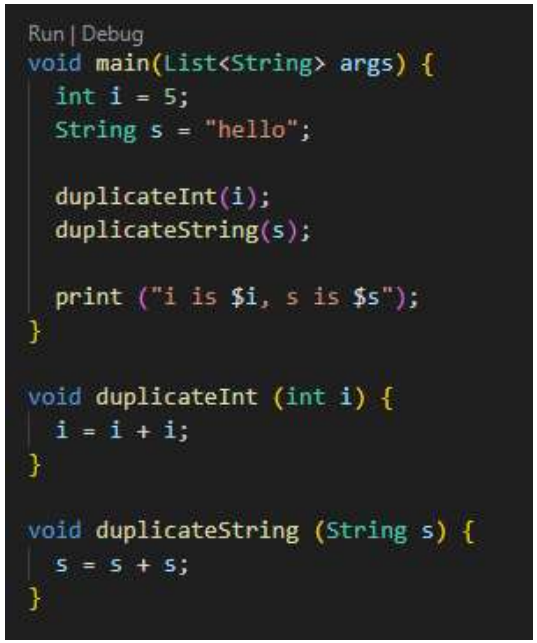
Call-by-value and call-by-reference in Dart

- Explain the difference between call-by-value and call-by-reference
- Know how to define a generic class
- Be able to use BoxedValue<T> class when a variable of the calling method should be modified by a called method

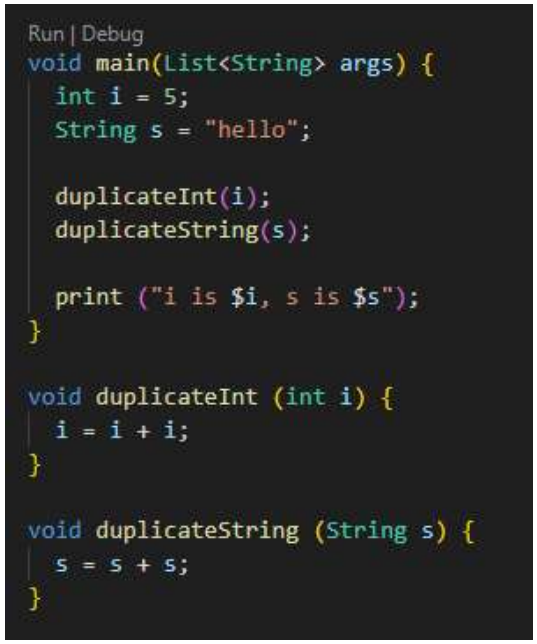
Quiz



```
Run | Debug
void main(List<String> args) {
  int i = 5;
  String s = "hello";

  duplicateInt(i);
  duplicateString(s);

  print ("i is $i, s is $s");
}

void duplicateInt (int i) {
  i = i + i;
}

void duplicateString (String s) {
  s = s + s;
}
```

What is the output ?

Solution



```
Run | Debug
void main(List<String> args) {
  int i = 5;
  String s = "hello";

  duplicateInt(i);
  duplicateString(s);

  print ("i is $i, s is $s");
}

void duplicateInt (int i) {
  i = i + i;
}

void duplicateString (String s) {
  s = s + s;
}
```

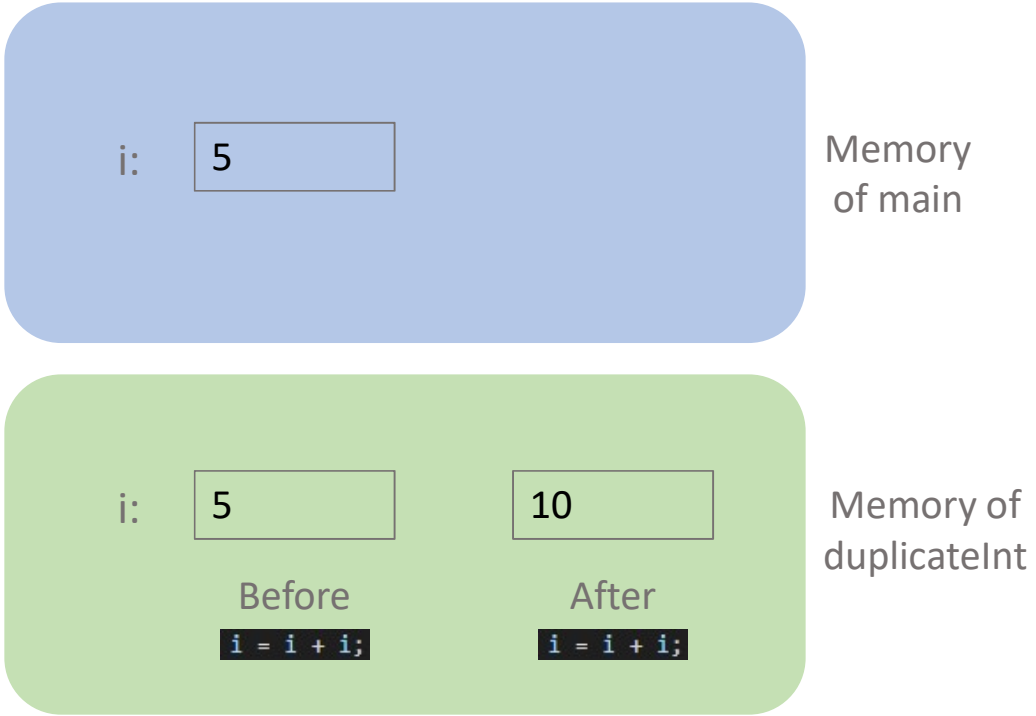
What is the output ?

```
i is 5, s is hello
```

What happens for duplicateInt in memory ?

```dart
void main(List<String> args) {
  int i = 5;
  String s = "hello";

  duplicateInt(i);
  duplicateString(s);

  print ("i is $i, s is $s");
}
```

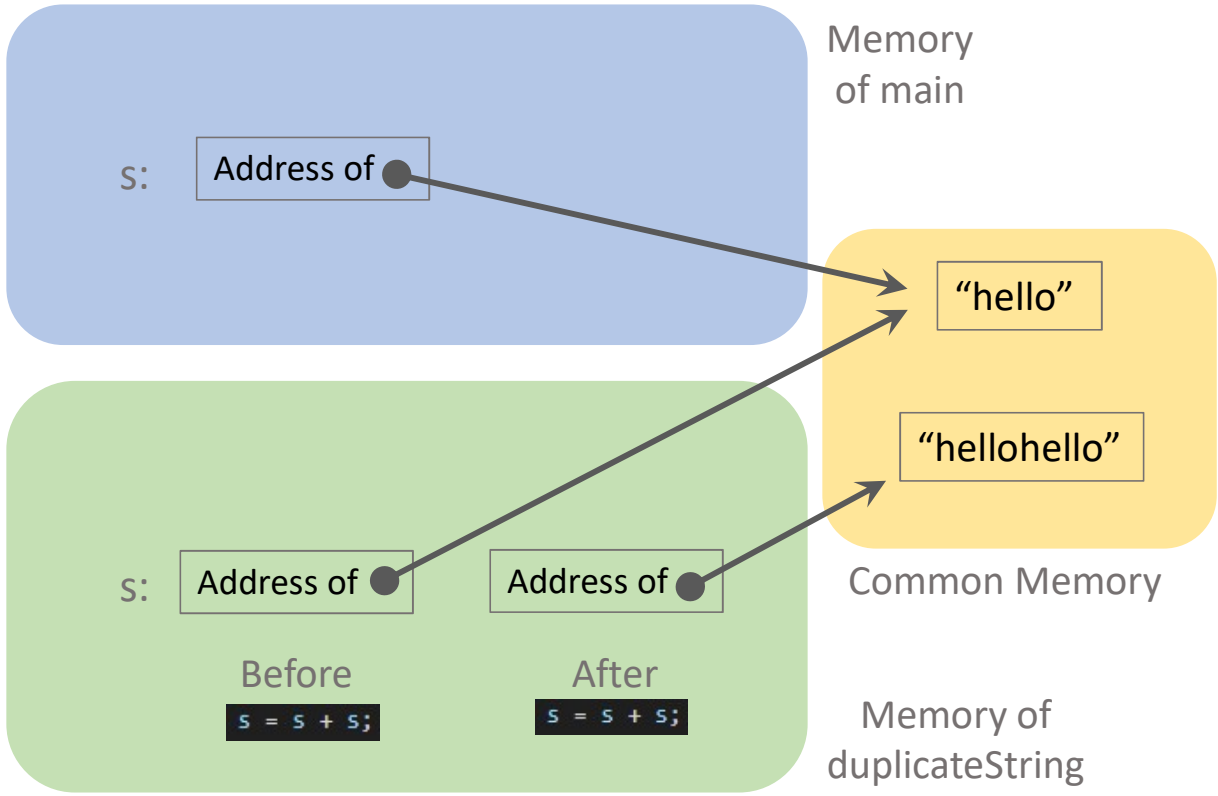
```dart
void duplicateInt (int i) {
  i = i + i;
}
```



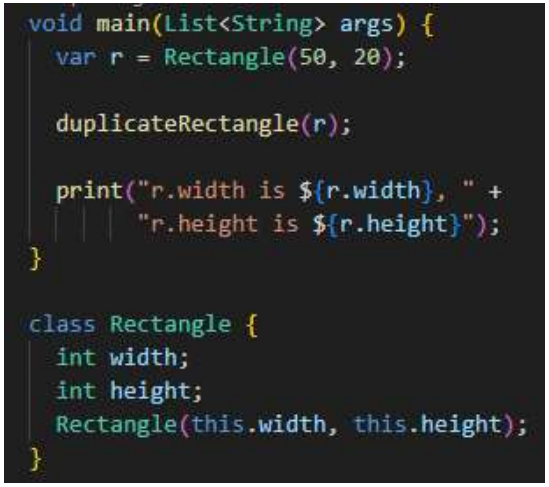
What happens for duplicateString in memory ?

```dart
void main(List<String> args) {
  int i = 5;
  String s = "hello";

  duplicateInt(i);
  duplicateString(s);

  print ("i is $i, s is $s");
}
```

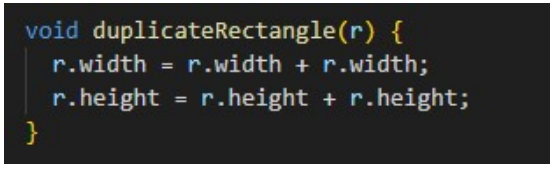
```dart
void duplicateString (String s) {
  s = s + s;
}
```



Quiz



```dart
void main(List<String> args) {
  var r = Rectangle(50, 20);

  duplicateRectangle(r);

  print("r.width is ${r.width}, " +
        "r.height is ${r.height}");
}

class Rectangle {
  int width;
  int height;
  Rectangle(this.width, this.height);
}
```



```dart
void duplicateRectangle(r) {
  r.width = r.width + r.width;
  r.height = r.height + r.height;
}
```

What is the output ?

Solution

```dart
void main(List<String> args) {
  var r = Rectangle(50, 20);

  duplicateRectangle(r);

  print("r.width is ${r.width}, " +
        "r.height is ${r.height}");
}

class Rectangle {
  int width;
  int height;
  Rectangle(this.width, this.height);
}
```

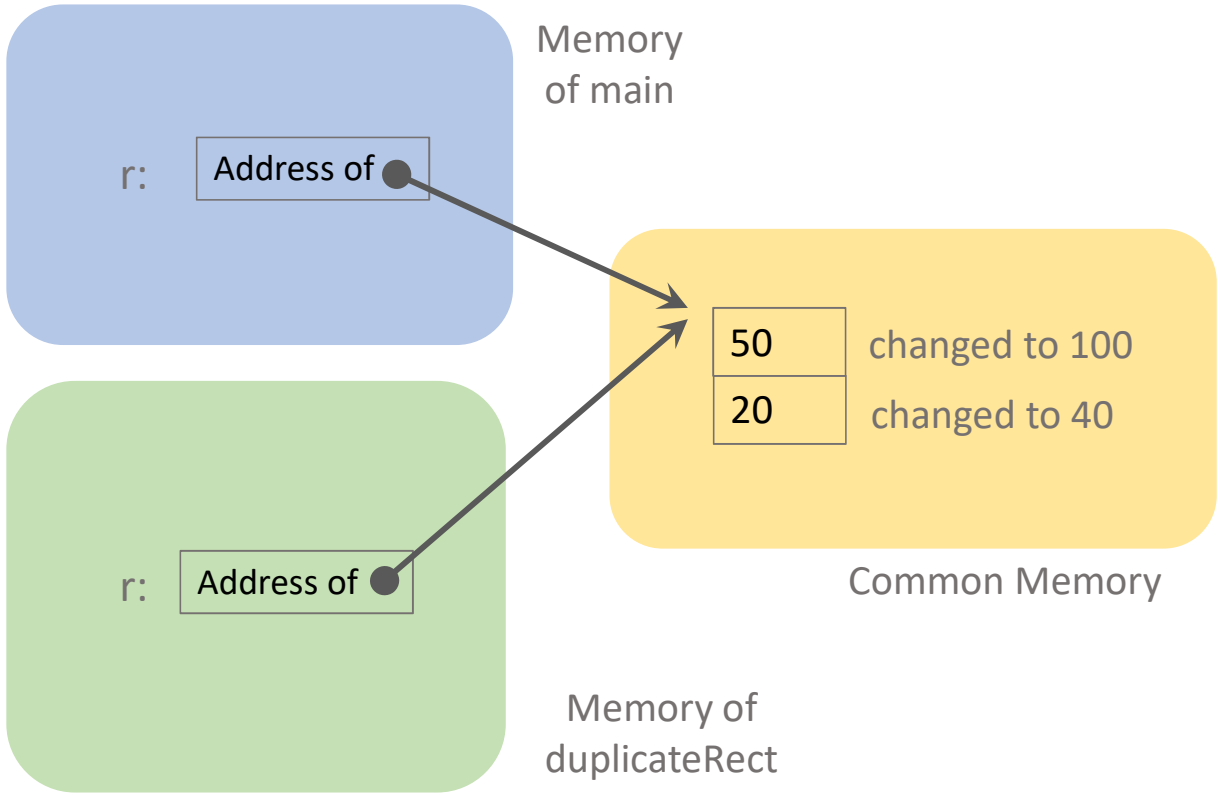
```dart
void duplicateRectangle(r) {
  r.width = r.width + r.width;
  r.height = r.height + r.height;
}
```

```
r.width is 100,
r.height is 40
```

What happens for duplicateRectangle in memory ?

```
void main(List<String> args) {
  var r = Rectangle(50, 20);

  duplicateRectangle(r);

  print("r.width is ${r.width}, " +
        "r.height is ${r.height}");
}

class Rectangle {
  int width;
  int height;
  Rectangle(this.width, this.height);
}
```

```
void duplicateRectangle(r) {
  r.width = r.width + r.width;
  r.height = r.height + r.height;
}
```



Alternative: Work with return values

Instead of:

```
Run | Debug
void main(List<String> args) {
  int i = 5;
  String s = "hello";

  duplicateInt(i);
  duplicateString(s);

  print ("i is $i, s is $s");
}

void duplicateInt (int i) {
  i = i + i;
}

void duplicateString (String s) {
  s = s + s;
}
```

Now with return values:

```
Run | Debug
void main(List<String> args) {
  int i = 5;
  String s = "hello";

  i = duplicateInt(i);
  s = duplicateString(s);

  print ("i is $i, s is $s");
}

int duplicateInt (int i) {
  return i + i;
}

String duplicateString (String s) {
  return s + s;
}
```

```
i is 10, s is hellohello
```

But there are situations in flutter where you cannot work with the return value.
We will see one in our next lesson “three-axis-transform and DRY principle”.

Quiz

```dart
void main(List<String> args) {
  var bi = BoxedInt(5);
  var bs = BoxedString("hello");

  duplicateBoxedInt(bi);
  duplicateBoxedString(bs);

  print ("${bi.value}, ${bs.value}");
}

void duplicateBoxedInt (BoxedInt bi) {
  bi.value = bi.value + bi.value;
}

void duplicateBoxedString (BoxedString bs) {
  bs.value = bs.value + bs.value;
}
```

```dart
class BoxedInt {
  BoxedInt(this.value);
  int value;
}

class BoxedString {
  BoxedString(this.value);
  String value;
}
```

What is the output ?

Solution

```
void main(List<String> args) {
  var bi = BoxedInt(5);
  var bs = BoxedString("hello");

  duplicateBoxedInt(bi);
  duplicateBoxedString(bs);

  print ("${bi.value}, ${bs.value}");
}

void duplicateBoxedInt (BoxedInt bi) {
  bi.value = bi.value + bi.value;
}

void duplicateBoxedString (BoxedString bs) {
  bs.value = bs.value + bs.value;
}
```

```
10, hellohello
```

```
class BoxedInt {
  BoxedInt(this.value);
  int value;
}

class BoxedString {
  BoxedString(this.value);
  String value;
}
```

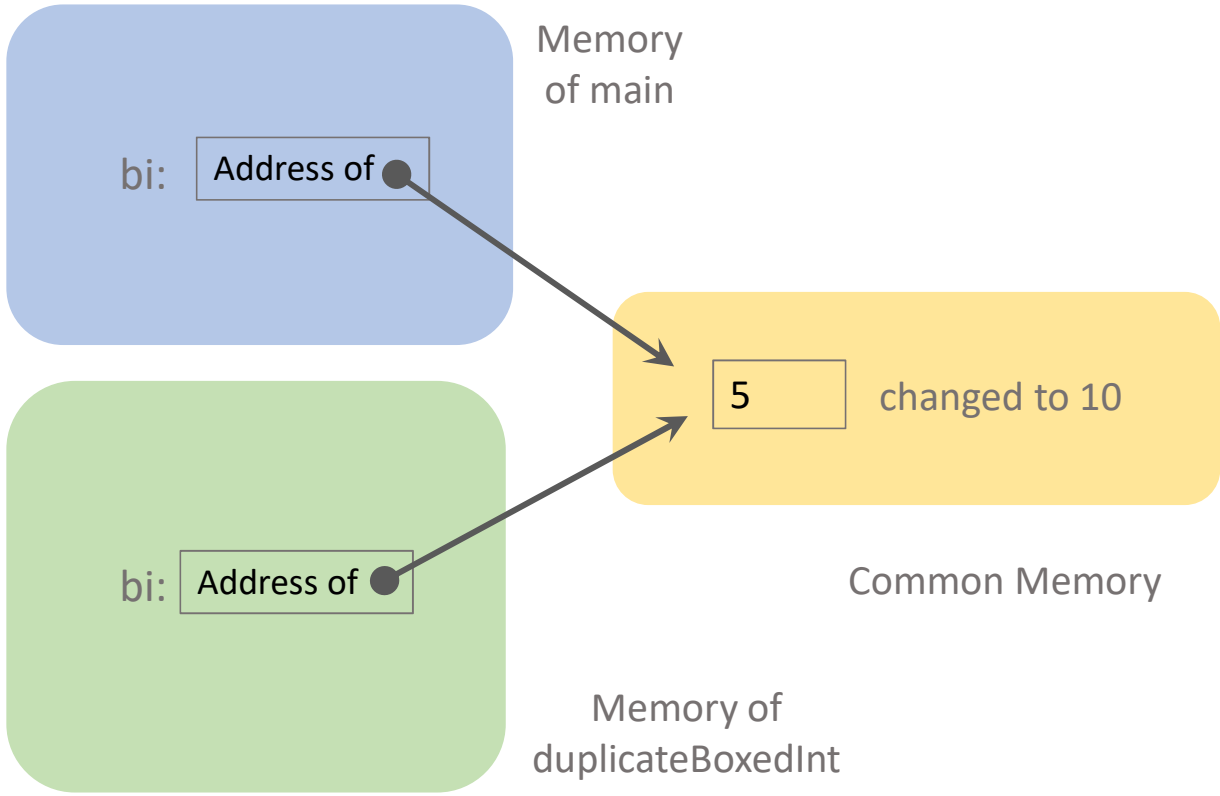
What happens for duplicateBoxedInt in memory ?

```
void main(List<String> args) {
  var bi = BoxedInt(5);

  duplicateBoxedInt(bi);

  print (bi.value);
}
```

```
10
```

```
class BoxedInt {
  BoxedInt(this.value);
  int value;
}
```

```
void duplicateBoxedInt (BoxedInt bi) {
  bi.value = bi.value + bi.value;
}
```



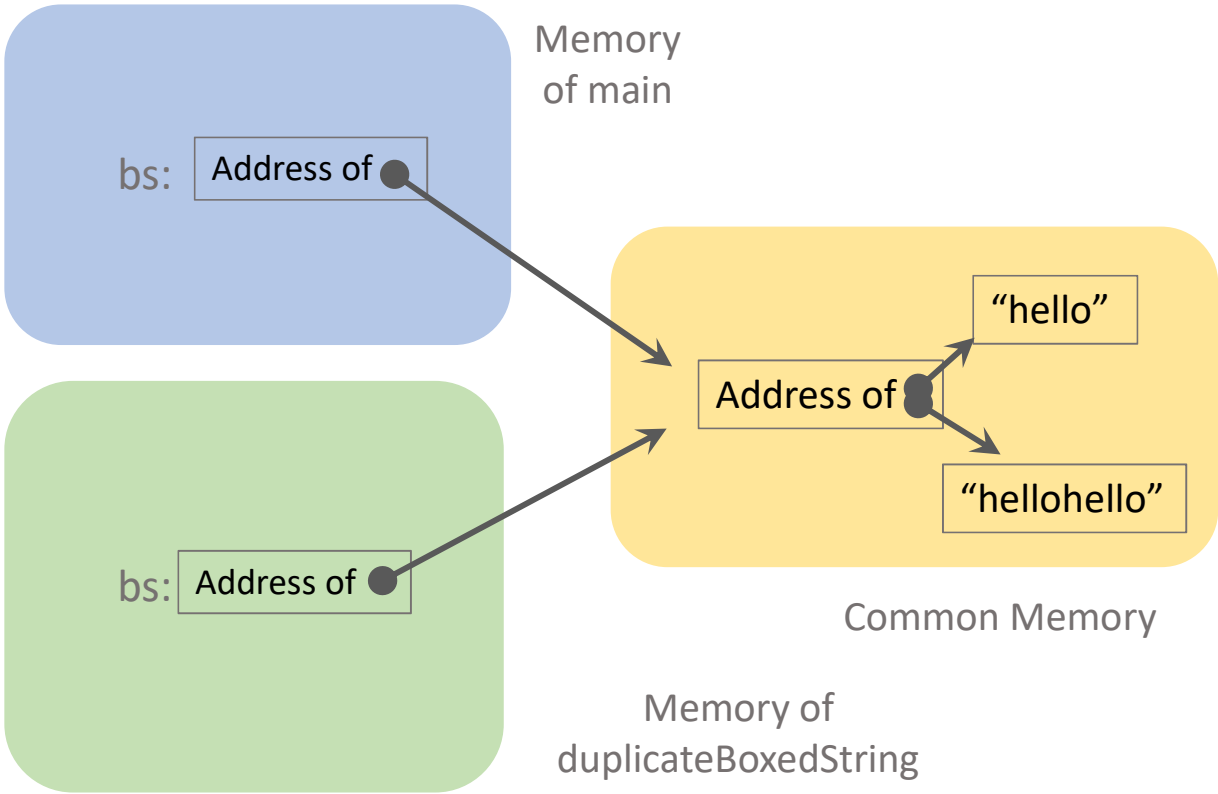
What happens for duplicateBoxedString in memory ?

```
hellohello
```

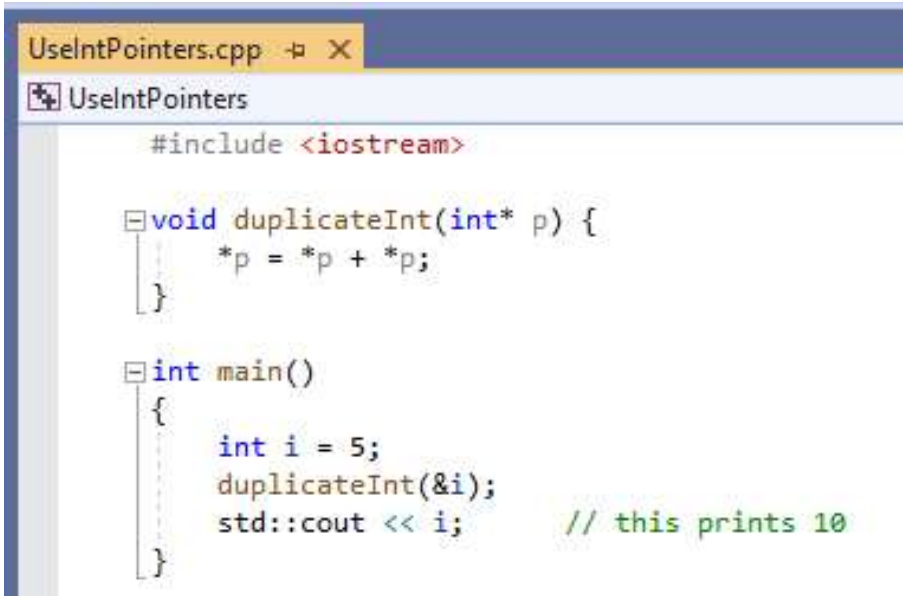
```
void main(List<String> args) {
  var bs = BoxedString("hello");

  duplicateBoxedString(bs);

  print (bs.value);
}
```

```
class BoxedString {
  BoxedString(this.value);
  String value;
}
```

```
void duplicateBoxedString (BoxedString bs) {
  bs.value = bs.value + bs.value;
}
```



Pointers in C++




```
class BoxedInt {
  BoxedInt(this.value);
  int value;
}
```

```
void duplicateBoxedInt (BoxedInt bi) {
  bi.value = bi.value + bi.value;
}
```

```
void main(List<String> args) {
  var bi = BoxedInt(5);

  duplicateBoxedInt(bi);

  print (bi.value);
}
```

Generic Type BoxedValue<T>

```dart
void main(List<String> args) {
  var bi = BoxedInt(5);
  var bs = BoxedString("hello");

  duplicateBoxedInt(bi);
  duplicateBoxedString(bs);

  print ("${bi.value}, ${bs.value}");
}

void duplicateBoxedInt (BoxedInt bi) {
  bi.value = bi.value + bi.value;
}

void duplicateBoxedString (BoxedString bs) {
  bs.value = bs.value + bs.value;
}

class BoxedInt {
  BoxedInt(this.value);
  int value;
}

class BoxedString {
  BoxedString(this.value);
  String value;
}
```

```dart
void main(List<String> args) {
  var bi = BoxedValue<int>(5);
  var bs = BoxedValue<String>("hello");

  duplicateBoxedInt(bi);
  duplicateBoxedString(bs);

  print ("${bi.value}, ${bs.value}");
}

void duplicateBoxedInt (BoxedValue<int> bi) {
  bi.value = bi.value + bi.value;
}

void duplicateBoxedString (BoxedValue<String> bs) {
  bs.value = bs.value + bs.value;
}
```

```dart
class BoxedValue<T> {
  BoxedValue(this.value);
  T value;
}
```

Reminder (copy from slide 33 in “07 Dart classes.pptx”):

List as a sample of a generic type

We used already one in our first hello.dart:

```dart
void main(List<String> args) {
  for (var arg in args) {
    print(arg);
  }
}
```

Other samples:

```dart
void testList() {
  List<int> intList = [1, 2, 3, 5, 7, 11, 13];
  List<User> userList = [User.withFullName("Franz Maier")];

  print("last in intList is ${intList.last}");
  print("intList is $intList");

  userList.add(PayingUser("Willi", "Zahn", "ibanWZ"));
  for (var user in userList) {
    print("$user");
  }
  userList.removeLast();
}
```

```
last in intList is 13
intList is [1, 2, 3, 5, 7, 11, 13]
```

```
User Franz Maier
PayingUser Willi Zahn ibanWZ
```