Congratulations! You passed!
Grade received 100%Latest Submission Grade 100%To pass 80% or higher

Go to next item

1. Which option represents the correct way to call the function 'sayHello' defined in the object below: 1 / 1 point

```
object Greeting {
 fun sayHello() = println("hello")
}
```

- (selected) **println(Greeting. sayHello())**
- **println(Greeting. sayHello)**
- **println(Greeting(). sayHello())**

Correct
Correct! You access a member of an object simply by using the object's name and the dot operator.

2. Which of these is correct if you wish to navigate from an activity 'SourceActivity' to another activity 'DestinationActivity' in Android? (Assume 'Context' object can be referenced using 'context' and instance of 'SourceActivity' can be referenced as 'sourceActivity'). Select all that apply. 1 / 1 point

- [x] **val intent = Intent(sourceActivity, DestinationActivity::class.java)**

Correct
Correct! You can define an intent by passing the instance of calling activity (as 'Activity' class inherits from 'Context' class) and class reference of the activity to be started.

- [x] **val intent = Intent(context, DestinationActivity::class.java)**

Correct
Correct! You can define an intent by passing the context object and class reference of the activity to be started.

- [ ] val intent = Intent(DestinationActivity::class.java, sourceActivity)
- [ ] **val intent = Intent(SourceActivity::class.java, DestinationActivity::class.java)**

3. You are asked to implement an extension function for a class named 'FoodItem' that would print out its ingredients field. How would the extension function look? 1 / 1 point

- (selected) **fun FoodItem.printIngredients() { println(ingredients) }**
- **fun FoodItem.printIngredients(ingredients: List<Ingredient>) { println(ingredients) }**
- **fun printIngredients(foodItem: FoodItem) { println(foodItem.ingredients) }**
- **fun List<Ingredient>.printIngredients(foodItem: FoodItem) { println(this) }**

Correct
Correct! This is the right syntax for the requested extension function.

4. When should you use mocks in your tests? 1 / 1 point

- (selected) When there are objects that are not to be tested but are needed because the code under test depends on them.
- When you need to define a complete alternate definition of an object to be used for testing.
- When you need to test only some specific behavior of an object

Correct
Correct! You use mocks to simulate behavior of objects that the test code depends on.

5. Once features and software requirements are planned, what is the next step in a test-driven development approach? 1 / 1 point

- Writing code to implement requirements
- Executing tests
- Refactoring code to fix errors
- (selected) Writing tests

Correct
Correct! The tests are written first such that they cover the scenarios of software application requirements. Later, code is written with the intent of passing the tests.

6. You need to instantiate a list of numbers. Which of the following statements are valid in Kotlin? 1 / 1 point

- **val numbers: List<Int> = [1, 4, 9]**
- **val numbers: List<Int> = (1, 4, 9)**
- (selected) **val numbers: List<Int> = listOf(1, 4, 9)**
- **val numbers: List<Int> = 1, 4, 9**

Correct
Correct! This is the correct syntax for instantiating a list of strings.

7. What is the output of this code: 1 / 1 point

```
val map = mapOf(
 1 to 90,
 2 to 93,
 3 to 91,
 4 to 93,
 2 to 95,
 5 to 93
)
println(map)
```

- (selected) {1=90, 2=95, 3=91, 4=93, 5=93}
- {1=90, 2=93, 3=91, 4=93, 2=95, 5=93}
- {1=90, 2=93, 3=91, 4=93, 5=93}

Correct
Correct! A map stores unique keys, but the values do not have to be unique.

8. Which of these represents a correct syntax of defining a generic class? 1 / 1 point

- (selected) **class Item<T>(t: T) { }**
- **class <T>.Item(t: T) { }**
- **class <T> Item(t: T) { }**

Correct
Correct. The generic parameter enclosed in the angle brackets is written after the class name.

9. Which of these below are higher-order functions? Select all that apply. 1 / 1 point

- [ ] **fun display(x: (Int)) -> Unit**
- [x] **fun display(): (Int) -> Unit**

Correct
Correct! This is a higher-order function as it returns a function.

- [ ] **fun display(x: Int) : Unit**
- [x] **fun display(x: (Int) -> Unit)**

Correct
Correct! This is a higher-order function as it takes another function as a parameter.

10. What will be the output of the following code? 1 / 1 point

```
val numberMap = mapOf(
 5 to 6,
 3 to 2,
 8 to 7,
 4 to 1
)
val output = numberMap.map { entry ->
 entry.value
},filter {
 it > 3
},fold(2) { x, y ->
 x + y
}
println(output)
```

- (selected) 15
- 3
- 17
- 13

Correct
Correct! You correctly computed the outputs of map, filter and fold functions in the above code.