Md Rahman

1. What is static, inferred and dynamic typing?

2. How does stringiest pools work in JAVA

3. What is functional Programming and how does it differ from pure object oriented?

4. What is lambda function?

5. What are some ways Kotlin achieves null safety?

1. What is static, inferred and dynamic typing?

Java has static typing, where a variable type must be first declared and then initialized. For example:

Int num;

num = 4;

In a dynamic typing, variable types do not have to be declared before they can be used, and their type can even change at runtime. Ex:

num = 5;

Kotlin has what’s known as inferred typing, where, like dynamic, the type of the variable is not set until you specify a value. However, unlike dynamic, the type can not be changed at runtime. So:

Var i = true; // compiler infers that i is type boolean

i= “xyz” // invalid

2. How does string pool work in Java?

A selection of strings stored in the heap, allowing for string constants to be reusable. For example:

String s = null dedicates a space for a string in memory but it’s empty. The variable s points to this space.

s=”hello” now the string “hello” will take up that space in memory.

s=”world” now, when accessing the value of s, “world” would print. However the previous string “hello” is still in that memory, it’s just inaccessible.

3. What is functional Programming and how does it differ from pure object oriented?

In all programs, there are two primary components: the data (what the program knows) and the behaviors (a means to manipulate that data). OOP brings that data and behaviors together in a single entity called an object. Functional programming however says that data and behavior should be kept separate for clarity.

OOP use classes to generate objects and their behaviors. We access these behaviors (methods) whenever we need to manipulate these objects. FP on the other hand prefers to not group this data and behavior in one place; rather, we have smaller methods that do a small part of a much larger job, and the combination of all these smaller methods is called a composition.

4. What is a lambda function?

Lambda’s are small, anonymous functions that can take any number of arguments but only have one expression. For example:

x = lambda a : a + 20

print(x(5));

A+20 is the expression assigned to this lambda. The first “a” after “lambda” is the argument. This will print 25.

5. What are some ways Kotlin achieves null safety?

Kotlin has tried eliminating the infamous “Null pointer exception.” NPEs are caused when trying to access data that isn’t there. In Kotlin, the type system will distinguish between references that are capable of holding null, and those that aren’t. A regular String variable cannot hold null. In order to do it, we must declare a nullable string. Ex:

Var s: String = “Hi”

s = null // compilation error

Var b: String? = “Hi”

b= null // acceptable

We also have safe calls, using a ?. operator. We use this when we’re not sure if a particular value exists. For example, if an employee Jim is possibly assigned to a department, return the name of that department head (if there is one):

Jim?.department?.head?.name //This is known as a chain, and will return null if any of the properties are null.

Another key word is “let”, which allows us to perform operations only on non null values.