Jamarcus Coulter

1. What is static, inferred and dynamic typing?

Static typed programming languages requires variables to be explicitly declared with a data type but does not have to a value defined to the variable. These types of programming languages perform type checking during compile time. If a variable data type and assigned value do not match, an error will be produced and generally be unable to be compiled.

Dynamic typed programming languages requires variables to be associated with a value but do not need to have the data type declared. Type checking is performed at runtime and assigns the data type of the variable when translated. This makes coding simpler and more likely to execute but can cause unwanted behaviors did to improper use or even typos.

Inferred typed programming languages is similar to dynamic typing in that variables do not have to data type declared for the variable. The difference is that inferred type checking is done at compile similar to static, but the compiler can infer the data type of the variable when compiling.

2. How does stringiest pools work in JAVA?

Due to String objects being immutable, JAVA has a special memory region allocated for Strings. JAVA does string interning. String interning is when a String variable is created the JVM checks the allocated pool for literal Strings for a match. If a match is found, the memory location is reference and returned for the string variable. If no match is found, the literal String is added to the pool and then referenced. If the new operator is used to create a String, a new object of the string is added to the pool with it own memory location.

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

Functional programming is based on treating computations as mathematical functions without states and mutable data, which eliminates side effects due to these properties. This programming paradigm is idempotent where a function with the same inputs will always return the same results. Flow control is done by direct function calls and the use of recursion. Functional programming is generally dynamic typed.

Functional programming differs from pure object oriented programming (OOP). OOP has states, mutable data, and has sides effect due to these. A function may produce a different result with the same values due to the current state of the system. OOP, while allows recursion, has control statements like while and for loops. OOP is generally static typed and focus on how to solve a problem, while functional programming focus is what to solve for the problem.

4. What is lambda function?

Lambda function is creating an anonymous function with any number of arguments but only a single expression is allowed. This expression is evaluated and return. Lambda functions can be used anywhere a single expression function can be used.

5. What are some ways Kotlin achieves null safety?

By default, Kotlin compiler will not allow a null reference. To create a null reference, a question mark is appended to the type definition and then to access the null reference an assertNull() function is used.

Safe calls are a more efficient way of handling null instead of a per case basics. let() is used to perform an action on a null reference parameter (it) and can be used with also() to perform additional operation if a null is found. run() works similar to let() but uses this operator perform an action within the body of a function.

The Elvis operator (?:) returns the value of an operation if null reference is found. The unsafe get (!!) will throw a null pointer exception if a null reference is found.

Some classes in Kotlin, like List, have a method filterNotNull() which will only return non-null references.