Correct answers are marked with an “🡸” after the answer.

Quizzes should be included **after** the lecture mentioned above it.

**10.variables**

Formularbeginn

Question 1:

**Which of the following is a correct creation of a variable?**

* 

variable language = "Kotlin"

* 

var language = "Kotlin" 🡸

* 

String language = "Kotlin"

Formularbeginn

Question 2:

**Which of the following is a correct creation of a value in Kotlin (that is, a final variable that cannot be reassigned).**

* 

value favorite = "Kotlin"

* 

final String favorite = "Kotlin"

* 

val favorite = "Kotlin" 🡸

Formularbeginn

Question 3:

**What is the difference between variables created using val vs. var?**

* 

Variables created using val are immutable, meaning that they cannot be reassigned a different value after their first assignment. X

* 

With val, we can only create number values. With var, we can create variables of any type.

**11.data-types**

Formularbeginn

Question 1:

**Which of the following is a correctly written data type in Kotlin?**

* 

integer

* 

Int 🡸

* 

Integer

Formularende

Formularbeginn

Question 2:

**Which of the following is a basic data type in Kotlin?**

* 

DoubleFloat

* 

Text

* 

String 🡸

Formularbeginn

Question 3:

**Which of the following is a correct assignment in Kotlin?**

* 

val i: Int = 42  🡸

* 

var a == 3

* 

variable n = 126

Formularbeginn

Question 4:

**Which of the following variables is mutable (that is, its value can be changed after the first assignment)?**

* 

val greeting = "Howdy!"

* 

var greeting: String = “Howdy!"  🡸

**12.expressions-statements**

Formularbeginn

Question 1:

**What is the difference between an expression and a statement?**

* 

Statements have a value, expressions do not.

* 

Expressions have a value, statements do not. 🡸

* 

There is no difference.

Formularbeginn

Question 2:

**Which of the following is a statement?**

* 

"Hey" + "there"

* 

5 \* (4 + 7)

* 

val greeting = "Hey there"  🡸

Formularbeginn

Question 3:

**Which of the following is an expression?**

* 

4.7 \* 11.8  🡸

* 

var str = "Expression"

**14.nullables**

Formularbeginn

Question 1:

**What is a nullable type?**

* 

We cannot create variables of nullable types.

* 

We can assign the value null to a variable of a nullable type. The null represents an empty reference (~”empty variable”) 🡸

Formularende

Formularbeginn

Question 2:

**Which is the following assignments creates a nullable variable?**

* 

var a = 3f

* 

val e = 2.71828

* 

var name: String? = "Kotlin"  🡸

* 

var email = null

Formularende

Formularbeginn

Question 3:

**How can we make a safe call on a nullable variable called "person"?**

* 

person.talk()

* 

person?.talk()  🡸

* 

person!!.talk()

Formularende

Formularbeginn

Question 4:

**Which of the following statements assures Kotlin that a nullable variable called "person" will not be null, so that Kotlin is going to unsafely call the function?**

* 

person?.talk()

* 

person->talk()

* 

person!!.talk()  🡸

Formularende

**15.if-statements**

Formularbeginn

Question 1:

**Which of the following code snippets uses the correct braces?**

* 
  1. if x < 4 {
  2. ...
  3. }
* 
  1. if [x < 4] {
  2. ...
  3. }
* 
  1. if (x < 4) {
  2. ...
  3. } 🡸

Formularende

Formularbeginn

Question 2:

**How can we specify many conditions?**

* 

Using:

* 1. elseif (condition) {
  2. // code
  3. }
* 

Using:

* 1. else if (condition) {
  2. code
  3. } 🡸
* 

Using:

* 1. else (condition) if {
  2. code
  3. }

Formularende

**16.when**

Formularbeginn

Question 1:

**Which of the following code snippets uses the correct syntax for when statements?**

* 
  1. when (variable) {
  2. if value => println("Quizzes are fun.")
  3. }
* 
  1. when variable {
  2. value -> println("Quizzes are fun.")
  3. }
* 
  1. when (variable) {
  2. value -> println("Quizzes are fun.")
  3. } 🡸

Formularende

**17.if-vs-when**

Formularbeginn

Question 1:

**Which of the following is a correctly written if statement in Kotlin?**

* 
  1. if 3 < 5 (
  2. println(“3 is less than 5”)
  3. )
* 
  1. if 4 == 5 {
  2. println(“4 is 5 in my universe”)
  3. }
* 
  1. if (2 < 3) {
  2. println(“Quizzes help me learn”)
  3. } 🡸

Formularende

Formularbeginn

Question 2:

**Which of the following is a correcty written when statement?**

* 
  1. when x (
  2. is 1 -> println(“x is one”)
  3. is 2 -> println(“x is two”)
  4. )
* 
  1. when (x) {
  2. 1 -> println(“x is one”)
  3. 2 -> println(“x is two”)
  4. else -> println(“x is neither one nor two”)
  5. } 🡸
* 
  1. when {x} {
  2. 1 => println(“x is one”)
  3. 2 => println(“x is two”)
  4. else => println(“x is neither one nor two”)
  5. }

Formularende

**19.advanced-when**

Formularbeginn

Question 1:

**Which of the following is a correct if expression?**

* 
  1. var y = if (x < 3) (
  2. "x is less than 3"
  3. ) else (
  4. "x is not less than three"
  5. )
* 
  1. var y = if (x < 3) {
  2. return "x is less than 3"
  3. } else {
  4. return "x is not less than three"
  5. }
* 
  1. var y = if (x < 3) {
  2. z = 7
  3. "x is less than 3"
  4. } else {
  5. "x is not less than three"
  6. } 🡸

Formularende

Formularbeginn

Question 2:

**Which of the following is a correct when expression?**

* 
  1. var s = when (x) {
  2. "Kotlin" -> "x is Kotlin"
  3. "Java" -> "x is Java"
  4. else -> "Maybe x is Python?"
  5. } 🡸
* 
  1. var s = when x {
  2. "Kotlin" -> println("x is Kotlin")
  3. "Java" -> println("x is Java")
  4. else -> println("Maybe x is Python?")
  5. }

Formularende

Formularbeginn

Question 3:

**Which is the following when statements makes correct use of ranges?**

* 
  1. when (x) {
  2. 11 to 99 -> println(“x is between 11 and 99”)
  3. else -> println(“x is not between 1 and 99”)
  4. }
* 
  1. when (x) {
  2. in 11..99 -> println(“x is between 11 and 99”)
  3. else -> println(“x is not between 1 and 99”)
  4. } 🡸

Formularende

Formularbeginn

Question 4:

**Which of the following when statements is written correctly?**

* 
  1. when (x) {
  2. is long -> println(“x is of type Long”)
  3. not in 5..20 -> println(“x is not between 5 and 20”)
  4. }
* 
  1. when (x) {
  2. is Long -> println(“x is of type Long”)
  3. !in 5..20 -> println(“x is not between 5 and 20”)
  4. } 🡸

Formularende

**23.arrays-vs-lists**

Formularbeginn

Question 1:

**How can we create an array in Kotlin?**

* 

var numbers = Array(1,2,3,4,5)

* 

var numbers = new Array(1,2,3,4,5)

* 

var numbers = intArrayOf(1,2,3,4,5)  🡸

Formularende

Formularbeginn

Question 2:

**How can we create a list in Kotlin?**

* 

var numbers = List(1,2,3,4,5)

* 

val mix = listOf(“Udemy”, 10, 3.5)  🡸

* 

var numbers = new ArrayList(1,2,3,4,5)

Formularende

Formularbeginn

Question 3:

**How can we retrieve the third element from an array or list variable called *languages*?**

* 

languages(2)

* 

languages[2]  🡸

* 

languages[3]

* 

languages{2}

Formularende

Formularbeginn

Question 4:

**How can we set the second element of an array or list called *magicNumbers* to 42?**

* 

magicNumbers[2] = 42

* 

magicNumbers(1) = 42

* 

magicNumbers[1] = 42  🡸

Formularende

Formularbeginn

Question 5:

**How can we get the length of an array or list called *states*?**

* 

states.length

* 

states.getLength()

* 

states.size  🡸

Formularende

**25.while**

Formularbeginn

Question 1:

**Which of the following is a correct for loop in Kotlin?**

* 
  1. for char in “Use the discussions!” (
  2. println(char)
  3. )
* 
  1. for (char in “I love Kotlin”) {
  2. println(char)
  3. } 🡸

Formularende

Formularbeginn

Question 2:

**Which of the following is a correct while loop in Kotlin?**

* 
  1. while [input != “quit”] (
  2. println(“You typed $input”)
  3. )
* 
  1. while (input != “quit”) {
  2. println(“You typed $input”)
  3. } 🡸

Formularende

Formularbeginn

Question 3:

**When reading a file line by line, which looping structure should you use?**

* 

A for loop, because I know how many iterations I need

* 

A while loop, because I don’t know how many lines there are in advance. I read line by line until the end of file (EOF) 🡸

Formularende

Formularbeginn

Question 4:

**When looping over an array or list, which looping structure is usually preferable?**

* 

A for loop, because I can easily loop over all its elements or even get the length of the list or array using array.size so I know how many iterations I need. 🡸

* 

A while loop, because I cannot know in advance how many iterations I will need

Formularende

**27.naming-loops**

Formularbeginn

Question 1:

**Which of the following code snippets uses break correctly to avoid unnecessary iterations after an ‘o’ was found in the string?**

* 
  1. for (char in "I love Kotlin") {
  2. if (char == 'o') {
  3. println("Found an o")
  4. }
  5. break
  6. }
* 
  1. for (char in "I love Kotlin") {
  2. break
  3. if (char == 'o') {
  4. println("Found an o")
  5. }
  6. }
* 
  1. for (char in "I love Kotlin") {
  2. if (char == 'o') {
  3. println("Found an o")
  4. break
  5. }
  6. } 🡸

Formularende

Formularbeginn

Formularbeginn

Question 2:

**Which of the following code snippets uses continue correctly to avoid doing further work on too short strings (for some purpose)?**

* 
  1. for (str in listOfNames) {
  2. if (str.length < 2) {
  3. // work only with longer strings...
  4. }
  5. continue
  6. }
* 
  1. for (str in listOfNames) {
  2. if (str.length < 2) {
  3. continue
  4. }
  5. // work only with longer strings...
  6. } 🡸
* 
  1. for (str in listOfNames) {
  2. continue
  3. if (str.length < 2) {
  4. // work only with longer strings...
  5. }
  6. }

Formularende

Formularbeginn

Question 3:

**How can we refer to a loop called @myLoop in a continue operator (break works the same way)?**

* 

continue @myLoop

* 

continue myLoop@

* 

continue@myLoop  🡸

Formularende

**29.congrats**

Formularbeginn

Question 1:

**Which of the following are all correct data types in Kotlin?**

* 

String, Integer, Short, Large

* 

Short, Char, Boolean, Triple

* 

Double, Char, Byte, Boolean 🡸

Formularende

Formularbeginn

Question 2:

**What is an immutable variable and how can we define it?**

* 

An immutable variable cannot be null and we create it using val.

* 

An immutable variable cannot be reassigned and we create one using val. 🡸

* 

An immutable variable cannot be reassigned and we create one using var.

Formularende

Formularbeginn

Question 3:

**What are nullable types and how can we create a nullable String variable?**

* 

Nullable types allow the assignment of the special null value; we can create a nullable String using var str: String? 🡸

* 

Nullable types allow the assignment of the special null value; we can create a nullable String using var str: String = null

* 

Nullable types can only be assigned to the null value; we can create a nullable String using var str: String?

Formularende

Formularbeginn

Question 4:

**What are the two keywords we can use in Kotlin for conditional execution?**

* 

if and switch

* 

when and switch

* 

if and when 🡸

Formularende

Formularbeginn

Question 5:

**In most cases, we will use *if*instead of *when*. But when do we use *when*instead of *if*?**

* 

In cases where we need to specify an arbitrary condition.

* 

If we want to switch execution behavior based on a set of distinct values a variable can have. 🡸

Formularende

Formularbeginn

Question 6:

**When do we use arrays over lists?**

* 

When we need to add and delete elements.

* 

When we want to store only numbers.

* 

When we have a fixed number of elements. 🡸

Formularende

Formularbeginn

Question 7:

**When do we use for loops over while loops?**

* 

Whenever we know the number of iterations in advance or want to loop over a collection. 🡸

* 

When we need to specify an arbitrary looping condition.

* 

Whenever we do NOT know how many iterations we will need.

Formularende

**31.functions**

Formularbeginn

Question 1:

**How can we create a function with a parameter of type Double?**

* 
  1. fun round(decimal: Double) {
  2. //...
  3. } 🡸
* 
  1. function round(decimal: Double) {
  2. // ...
  3. }
* 
  1. fun round(Double decimal) {
  2. // ...
  3. }

Formularende

Formularbeginn

Question 2:

**Which of the following functions returns a value of type Boolean?**

* 

fun isBoosted(b: Boolean) { ... }

* 

Boolean fun isBoosted() { ... }

* 

fun isBoosted(): Boolean { ... }  🡸

Formularende

Formularbeginn

Question 3:

**How can we call the following function inside our main function?**

1. **fun square(x: Double): Double {**
2. **return x\*x**
3. **}**

* 

val square = square()

* 

val square = 13.37.square()

* 

val square = square(13.37)  🡸

Formularende

**33.intro-oo**

Formularbeginn

Question 1:

**Why should we care about object-orientation?**

* 

It is used by virtually any big programming language

* 

It allows us to understand the OO principles behind languages like Java, Kotlin, Python, C#, C++, Scala, Smalltalk and hundreds more

* 

It is an intuitive way to develop software by modeling the real-world as objects

* 

All of the above. 🡸

Formularende

Formularbeginn

Question 2:

**Which of the following structures can a class contain in its body?**

* 

Methods, to hold functionality

* 

Properties, to hold data

* 

All of the above 🡸

Formularende

Formularbeginn

Question 3:

**How can we think of a class?**

* 

A class is a specific instantiation of a real-world object.

* 

A class defines a contract for other classes to implement, such as Driveable.

* 

A class is like a blueprint from which we can instantiate objects. 🡸

Formularende

**35.methods**

Formularbeginn

Question 1:

**How can we add a property to a class?**

* 

We have to use the property keyword: property something = "I am now a property"

* 

Just like we create variables outside classes: var something = "I am now a property" 🡸

* 

We use the fun keyword: fun something = "I am now a property"

Formularende

Formularbeginn

Question 2:

**How can we add a method called "fly" to a class?**

* 

fun fly() { ... }  🡸

* 

method fly() { ... }

* 

fun fly { ... }

Formularende

Formularbeginn

Question 3:

**How can we add a method called "prepare" with one Double and one Boolean parameter?**

* 

fun prepare(Double d, Boolean b) { ... }

* 

fun prepare[d: Double, b: Boolean] { ... }

* 

fun prepare(d: Double, b: Boolean) { ... }  🡸

Formularende

Formularbeginn

Question 4:

**How can we add a "contains" method that returns a Boolean to a class?**

* 

fun contains(c: Char): Boolean { ... }  🡸

* 

fun contains(c: Char) returns Boolean { ... }

* 

fun contains(c: Char) { return Boolean }

Formularende

**36.constructors**

Formularbeginn

Question 1:

**How can we create objects of a class called Person with no constructor parameters?**

* 

val person = Person

* 

val person = new Person()

* 

val person = Person()  🡸

Formularende

Formularbeginn

Question 2:

**How can we create objects of a type/class called Book with two String parameters title and author?**

* 

val book = new Book("Design Patterns", "Gang of Four")

* 

var book = Book("Design Patterns", "Gang of Four")  🡸

* 

var book == Book("Design Patterns", "Gang of Four")

Formularende

**38.named-parameters**

Formularbeginn

Question 1:

**How can we create an object of a class Laptop with two parameters named name and price?**

* 

val laptop = Laptop(price==699.99, name=="NoNameLaptop")

* 

val laptop = Laptop(price=699.99, name="NoNameLaptop")  🡸

* 

val laptop = new Laptop(price=699.99, name="NoNameLaptop")

Formularende

Formularbeginn

Question 2:

**What is the main advantage of using named parameters?**

* 

We remember the names of each parameter.

* 

We have to write more code for each call, thus making it less concise.

* 

When passing many unclear parameters, the code is more readable because the parameter names tell us what each value means. 🡸

Formularende

Formularbeginn

Question 3:

**How can we define a default value for a function parameter?**

* 

fun split(str: String, separator: Char = ' ') { ... }  🡸

* 

fun split(str: String, separator: Char default ' ') { ... }

* 

fun split(str: String, separator: Char) { separator = ' ' ... }

Formularende

Formularbeginn

Question 4:

**What do default values allow us to do in a very concise way?**

* 

Naming parameters.

* 

Overloading classes.

* 

Overloading constructors and functions. 🡸

Formularende

**39.inheritance**

Formularbeginn

Question 1:

**What can we do with open classes, in contrast to normal classes?**

* 

We can inherit from them to inherit its properties and methods in a child class and avoid duplicate code. 🡸

* 

We can access this class even from other packages.

* 

We can add abstract methods to them.

Formularende

Formularbeginn

Question 2:

**How can we create an open class?**

* 

class ExtendMe { ... }

* 

public class ExtendMe { ... }

* 

open class ExtendMe { ... }  🡸

Formularende

**40.abstract-classes**

Formularbeginn

Question 1:

**What can we use abstract classes for?**

* 

To define the differences between different classes.

* 

To encapsulate similarities between different classes and gather them all in one place to avoid code duplication. 🡸

Formularende

Formularbeginn

Question 2:

**What kind of properties and methods can we add to abstract classes?**

* 

We can add abstract, open and also normal properties and methods. 🡸

* 

We can add only abstract properties and methods.

* 

We can add only abstract and open properties and methods.

Formularende

Formularbeginn

Question 3:

**How can we create an abstract class in Kotlin?**

* 

abstraction class AbstractBeing { ... }

* 

abstract class AbstractBeing { ... }  🡸

* 

class abstract AbstractBeing { ... }

Formularende

**41.open-vs-abstract**

Formularbeginn

Question 1:

**How can we think of open classes vs. abstract classes?**

* 

Open classes MUST be inherited from (otherwise they are useless); abstract classes CAN be inherited from

* 

Open classes CAN be inherited from; abstract classes MUST be inherited from (otherwise they are useless) 🡸

* 

Open classes CAN be inherited from; abstract classes also CAN be inherited and can also inherit from any other class

Formularende

Formularbeginn

Question 2:

**How can we think of open vs. abstract properties and methods?**

* 

Open properties and methods MUST be overriden whereas abstract ones CAN be overriden in all child classes.

* 

There is no difference between open and abstract properties and methods.

* 

Open properties and methods are ALLOWED to be overriden whereas abstract ones MUST be overriden in all (non-abstract) child classes. 🡸

Formularende

Formularbeginn

Question 3:

**Do you think we can also add abstract methods to open (non-abstract) classes? (You may also try it out.)**

* 

Yes!

* 

No! 🡸

Formularende

**42.interfaces**

Formularbeginn

Question 1:

**How can we think of interfaces?**

* 

Interfaces are like containers for properties and methods.

* 

Interfaces define a contract that different classes may choose to follow. To do that, they have to override each method and property defined in the interface. 🡸

* 

Interfaces are like objects with state.

Formularende

Formularbeginn

Question 2:

**What would be a common name for an interface?**

* 

Learnable.

* 

Readable.

* 

Breakable.

* 

All of the above. 🡸

Formularende

Formularbeginn

Question 3:

**How can we create an interface Learnable with a learn() method in Kotlin?**

* 
  1. interface Learnable {
  2. fun learn()
  3. } 🡸
* 
  1. abstract interface Learnable {
  2. fun learn()
  3. }
* 
  1. interface Learnable {
  2. abstract fun learn()
  3. }

Formularende

Formularbeginn

Question 4:

**Can one class implement more than one interface?**

* 

Yes! 🡸

* 

No!

Formularende

**42.interfaces (again, as a conclusing quiz for the section)**

Formularbeginn

Question 1:

**Which of the following provides the highest level of abstraction?**

* 

A class.

* 

An abstract class.

* 

An interface. 🡸

Formularende

Formularbeginn

Question 2:

**To which of the following structures can we add properties and methods?**

* 

To classes.

* 

To abstract classes.

* 

To interfaces.

* 

To all of the above. 🡸

Formularende

Formularbeginn

Question 3:

**What is inheritance?**

* 

Inheritance means inheriting only the state from a parent class, meaning the properties and their values.

* 

Inheritance means inheriting properties and methods from a parent class to a new class we define so that we don't need to re-implement them all. 🡸

* 

Inheritance refers exclusively to interfaces. A class can inherit from an interface by implementing it.

Formularende

**43.overriding**

Formularbeginn

Question 1:

**An overriden method is, by default, final. So we cannot override it in again another child class.**

* 

True.

* 

False. 🡸

Formularende

Formularbeginn

Question 2:

**How can we prevent an overriden method from being overriden again in another (grand-)child class?**

* 

We can just declare it as override as normal: override fun someMethod() { ... }

* 

By explicitly making the method final: override final fun someMethod() { ... } 🡸

Formularende

Formularbeginn

Question 3:

**We can inherit from a parent class and also implement interfaces with one and the same class.**

* 

Yes! 🡸

* 

No!

Formularende

Formularbeginn

Question 4:

**How can we call a transaction() method from a parent class or interface from within a child class?**

* 

super.transaction()  🡸

* 

parent.transaction()

* 

TransactionClass.transaction()

Formularende

Formularbeginn

Question 5:

**If we inherit a method called switch() from a parent class LightSwitch and also an implemented interface Switch (which defines an implementation for the method), how can we call the switch() method from the LightSwitch class inside a child class?**

* 

super.switch()

* 

super<Switch>.switch()

* 

super<LightSwitch>.switch()🡸

Formularende

**44.data-classes**

Formularbeginn

Question 1:

**Which of the following do data classes take care of for us?**

* 

Defining a useful string representation of the class' objects using toString()

* 

Implementing a useful equals() method that compares all the properties

* 

Implementing a copy() method that allows us to easily make copies of a data object

* 

Allowing us to use the data objects with data types that use hashing, such as HashSet

* 

Allowing us to decompose data objects into their properties

* 

All of the above. 🡸

Formularende

Formularbeginn

Question 2:

**How can we create a Customer data class in Kotlin?**

* 

class DataCustomer(val name: String, val address: Address) { ... }

* 

data class Customer(val name: String, val address: Address) { ... }  🡸

* 

class data Customer(val name: String, val address: Address) { ... }

Formularende

Formularbeginn

Question 3:

**How can we create a copy with a different name of a data object called customer?**

* 

val copy = customer.clone(name = "Peter")

* 

val copy = customer.copy("Peter")

* 

val copy = customer.copy(name = "Peter")  🡸

Formularende

**47.enums**

Formularbeginn

Question 1:

**Enums should be preferred to lists of String or Int constants (if these define different possible values of a certain type, e.g. cardinal directions).**

* 

Yes! 🡸

* 

No!

Formularende

Formularbeginn

Question 2:

**Which of the following types/classes would be a good candidate for an enum?**

* 

Complex numbers

* 

Font styles (e.g. bold or italic) 🡸

* 

Books

Formularende

Formularbeginn

Question 3:

**How do you create an enum in Kotlin?**

* 

enum FontStyle { ... }

* 

class enum FontStyle { ... }

* 

enum class FontStyle { ... }  🡸

Formularende

**48.packages**

Formularbeginn

Question 1:

**What are packages used for?**

* 

To communicate across devices.

* 

To structure the project by combining source files with similar responsibilities into the same package. 🡸

* 

To add namespaces to every class and top-level function.

Formularende

Formularbeginn

Question 2:

**Which of the following would you choose as a (nested) package name?**

* 

some.stuff

* 

Com.Google.ProjectName

* 

com.kotlinrocks.model 🡸

Formularende

**49.imports**

Formularbeginn

Question 1:

**How can you import the Application class from JavaFX?**

* 

import javafx.application.Application  🡸

* 

import Application from javafx.application

* 

from javafx.application import Application

Formularende

Formularbeginn

Question 2:

**How can you import all classes from javafx.scene.control ?**

* 

import javafx.scene.control.\_

* 

import javafx.scene.control

* 

import javafx.scene.control.\*  🡸

Formularende

Formularbeginn

Question 3:

**Which of these can not be imported?**

* 

A whole enum class.

* 

A method from a class. 🡸

* 

A method from an object.

Formularende

Formularbeginn

Question 4:

**Which of these can you import?**

* 

Top-level functions. 🡸

* 

Functions nested inside top-level functions.

* 

Functions nested inside classes.

Formularende

**50.hexadecimal**

Formularbeginn

Question 1:

**Which decimal number is equal to the hexadecimal number 0x34 ?**

* 

832

* 

48

* 

52 🡸

* 

340

Formularende

Formularbeginn

Question 2:

**How do you represent the decimal number 23 in the hexadecimal system?**

* 

0x17 🡸

* 

0x15

* 

0x71

* 

0xAG

Formularende

Formularbeginn

Question 3:

**Which of the following colors contains the most green?**

* 

0xCCBBAA

* 

0x0000FF

* 

0xF0F0F0 🡸

* 

0x0F0F0F

Formularende

**51.binary-numbers**

Formularbeginn

Question 1:

**Which decimal number is represented by 0b101101 ?**

* 

90

* 

47

* 

45 🡸

Formularende

Formularbeginn

Question 2:

**Give the binary representation of the decimal number 248.**

* 

0b1111\_1000 🡸

* 

0b11111100

* 

0b1111\_1110

* 

0b1111\_1111

Formularende

Formularbeginn

Question 3:

**Which binary number is equal to the hex value 0xA3?**

* 

0b10100000

* 

0b11000001

* 

0b10100011 🡸

* 

0b00111011

Formularende

Formularbeginn

Question 4:

**How many bits do you need to represent the hex number #C7E?**

* 

4

* 

8

* 

12 🡸

Formularende

**53.information-hiding**

Formularbeginn

Question 1:

**What does information hiding mean?**

* 

Hiding the well-defined interface of your class.

* 

Restricting access from the outside to the internals of a class. 🡸

Formularende

Formularbeginn

Question 2:

**What's an advantage of information hiding?**

* 

Having more control over the possible values variables and objects used internally (inside your class).

* 

Being able to enforce invariants (characteristics of your class) more effectively.

* 

Better knowing how your class will be used from the outside (e.g. via which methods).

* 

All of the above. 🡸

Formularende

**54.getters-setters**

Formularbeginn

Question 1:

**In Kotlin, getters and setters only exist if specified explicitly.**

* 

Wrong! 🡸

* 

True!

Formularende

Formularbeginn

Question 2:

**How can you override the default getter and setter of a property?**

* 
  1. var prop: Int
  2. fun getProp() { ... }
  3. fun setProp(value) { ... }
* 
  1. var prop: Int
  2. getProp() { ... }
  3. setProp(value) { ... }
* 
  1. var prop: Int
  2. get() { ... }
  3. set(value) { ... } 🡸

Formularende

Formularbeginn

Question 3:

**Is the following code correct?**

1. **val fixed: String? = ""**
2. **get() = field**
3. **set(value) {**
4. **field = value**
5. **}**

* 

Nope. 🡸

* 

Yep.

Formularende

**55.visibilities**

Formularbeginn

Question 1:

**What are visibilities useful for?**

* 

To enforce the principle of information hiding.

* 

To define which properties and methods of a class can be accessed from which other places.

* 

To keep the internals of a class private and only release a well-defined interface to the outside.

* 

All of the above. 🡸

Formularende

Formularbeginn

Question 2:

**Which visibilities are available in Kotlin?**

* 

private  
inherited  
module  
public

* 

private  
protected  
package-private  
public

* 

private  
protected  
internal  
public 🡸

Formularende

Formularbeginn

Question 3:

**What does private visibility mean for a property or method of a class?**

* 

It can only be accessed inside that class. 🡸

* 

It can only be accessed inside that class and its subclasses.

* 

It can only be accessed from classes inside the same module.

* 

It can only be accessed from any class.

Formularende

Formularbeginn

Question 4:

**For which members of a class do you normally use public?**

* 

For properties and methods that should be overriden in subclasses.

* 

For properties and methods that form the well-defined interface of the class which should be exported to the outside. 🡸

* 

For properties and methods that shouldn't be accessed from the outside.

Formularende

Formularbeginn

Question 5:

**What's the default visibility?**

**Meaning, what would be the visibility of the property a  in:**

1. **class C {**
2. **val a = 42 // no explicit visibility defined**
3. **}**

* 

private

* 

protected

* 

internal

* 

Public 🡸

Formularende

**56.generics**

Formularbeginn

Question 1:

**What's an important principle in programming?**

* 

WET (Write errors throughout)

* 

HUMID (Have ur methods implemented directly)

* 

DRY (Don't repeat yourself) 🡸

Formularende

Formularbeginn

Question 2:

**What are generic classes used for?**

* 

To avoid code duplication (following DRY) by parametrizing a class with generic type parameters. 🡸

* 

To avoid code duplication (following DRY) by subtyping a base class.

* 

To avoid code duplication (following DRY) by using generic properties and methods.

Formularende

Formularbeginn

Question 3:

**How do you declare a generic class with one generic type parameter T?**

* 

<T> class Timeline<T> { ... }

* 

class Timeline<T> { ... }  🡸

* 

class <T> Timeline<T> { ... }

Formularende

**58.generic-functions**

Formularbeginn

Question 1:

**How do you declare a generic function?**

* 

fun <T> timelineOf(values: Array<T>): Timeline<T> { ... }  🡸

* 

fun timelineOf(values: Array<T>): Timeline<T> { ... }

* 

fun <T> timelineOf(values: Array<E>): Timeline<K> { ... }

Formularende