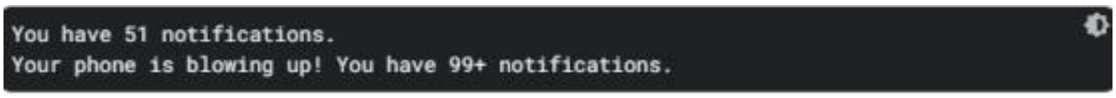
ASSIGNMENT - Code Lab Kotlin Program

Q1. Mobile notifications:

Typically, your phone provides you with a summary of notifications.
Write a program that prints the summary message based on the number of notifications that you received. The message should include:

1. The exact number of notifications when there are less than 100 notifications.
2. 99+ as the number of notifications when there are 100 notifications or more.

Complete the printNotificationSummary() function so that the program prints these lines:

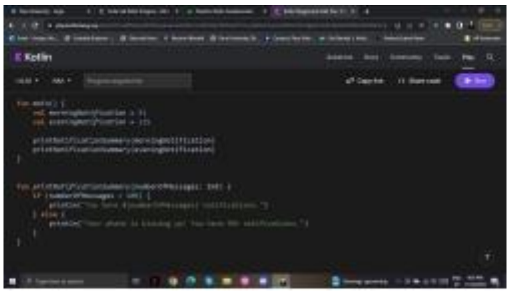


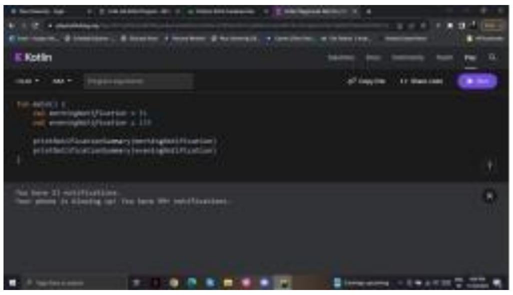
```
You have 51 notifications.
Your phone is blowing up! You have 99+ notifications.
```

ANSWSER:
Link: https://pl.kotl.in/dBpVfUESb

```
fun main() {
    val morningNotification = 51
    val eveningNotification = 135

    printNotificationSummary(morningNotification)
    printNotificationSummary(eveningNotification)
}



fun printNotificationSummary(numberOfMessages: Int) {
    if (numberOfMessages < 100) {
        println("You have ${numberOfMessages} notifications.")
    } else {
        println("Your phone is blowing up! You have 99+ notifications.")
    }
}
```

- The solution uses an if/else statement to print the appropriate notification summary message based on the number of notification messages received


[illegible]

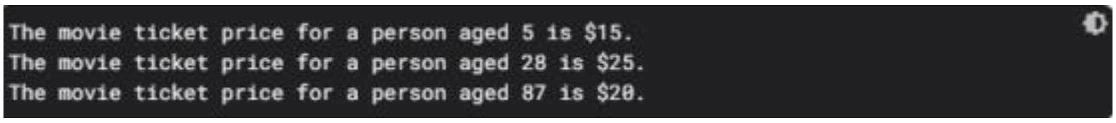

[illegible]

Q2. Movie Ticket Price:

Movie tickets are typically priced differently based on the age of moviegoers. Write a program that calculates these age-based ticket prices:

1. A children's ticket price of $15 for people 12 years old or younger.
2. A standard ticket price of $30 for people between 13 and 60 years old. On Mondays, discount the standard ticket price to $25 for this same age group.
3. A senior ticket price of $20 for people 61 years old and older. Assume that the maximum age of a moviegoer is 100 years old.

A -1 value to indicate that the price is invalid when a user inputs an age outside of the age specifications.
Complete the ticketPrice() function so that the program prints these lines:

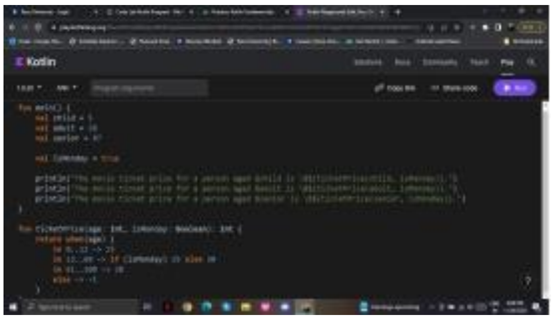


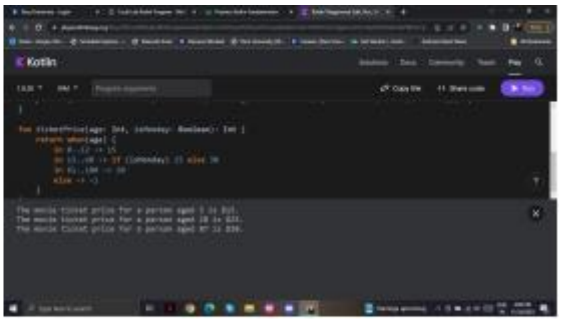
```
The movie ticket price for a person aged 5 is $15.
The movie ticket price for a person aged 28 is $25.
The movie ticket price for a person aged 87 is $20.
```

ANSWER:
Link: https://pl.kotl.in/64wk8kldh

```
fun main() {
    val child = 5
    val adult = 28
    val senior = 87

    val isMonday = true

    println("The movie ticket price for a person aged $child is \$${ticketPrice(child, isMonday)}.")
    println("The movie ticket price for a person aged $adult is \$${ticketPrice(adult, isMonday)}.")
    println("The movie ticket price for a person aged $senior is \$${ticketPrice(senior, isMonday)}.")
}

fun ticketPrice(age: Int, isMonday: Boolean): Int {
    return when(age) {
        in 0..12 -> 15
        in 13..60 -> if (isMonday) 25 else 30
        in 61..100 -> 20
        else -> -1
    }
}
```

-The solution uses a when expression to return the appropriate ticket price based on the moviegoer's age. It also uses a simple if/else expression for one of the when expression's branches to add the additional condition for the standard ticket pricing. The ticket price in the else branch returns a -1 value, which indicates that the price set is invalid for the else branch. A better implementation is for the else branch to throw an exception. You learn about exception handling in future units.





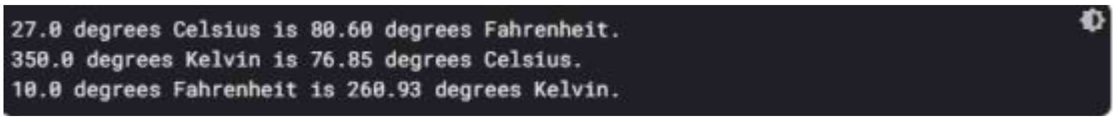
```
The movie ticket price for a person aged 5 is $15.
The movie ticket price for a person aged 28 is $25.
The movie ticket price for a person aged 87 is $20.
```

Q3. Temperature converter

There are three main temperature scales used in the world: Celsius, Fahrenheit, and Kelvin.
Write a program that converts a temperature from one scale to another with these formulas:

Celsius to Fahrenheit: ° F = 9/5 (° C) + 32
Kelvin to Celsius: ° C = K - 273.15
Fahrenheit to Kelvin: K = 5/9 (° F - 32) + 273.15

Note that the String.format("%.2f", /* measurement */) method is used to convert a number into a String type with 2 decimal places.
Complete the main() function so that it calls the printFinalTemperature() function and prints the following lines. You need to pass arguments for the temperature and conversion formula. Hint: you may want to use Double values to avoid Integer truncation during division operations.

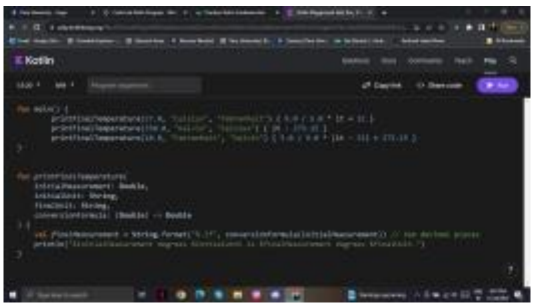


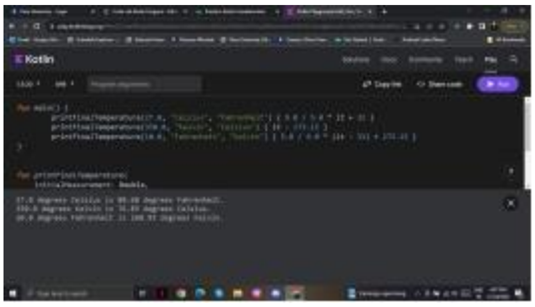
```
27.0 degrees Celsius is 80.60 degrees Fahrenheit.
350.0 degrees Kelvin is 76.85 degrees Celsius.
10.0 degrees Fahrenheit is 260.93 degrees Kelvin.
```

ANSWER:
Link: https://pl.kotl.in/H-WPDN7Po

```
fun main() {
    printFinalTemperature(27.0, "Celsius", "Fahrenheit") { 9.0 / 5.0 * it + 32 }
    printFinalTemperature(350.0, "Kelvin", "Celsius") { it - 273.15 }
    printFinalTemperature(10.0, "Fahrenheit", "Kelvin") { 5.0 / 9.0 * (it - 32) + 273.15 }
}
fun printFinalTemperature(
  initialMeasurement: Double,
  initialUnit: String,
  finalUnit: String,
  conversionFormula: (Double) -> Double
) {
  val finalMeasurement = String.format("%.2f", conversionFormula(initialMeasurement)) // two
decimal places
  println("$initialMeasurement degrees $initialUnit is $finalMeasurement degrees $finalUnit.")
}
```

-The solution requires you to pass a function as a parameter to the printFinalTemperature() function. The most succinct solution passes lambda expressions as the arguments, uses the it parameter reference in place of the parameter names, and makes use of trailing lambda syntax.



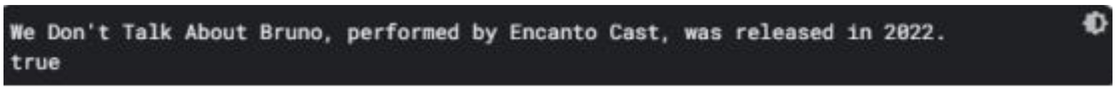


Q4. Song Catalog

Imagine that you need to create a music-player app.

Create a class that can represent the structure of a song. The Song class must include these code elements:

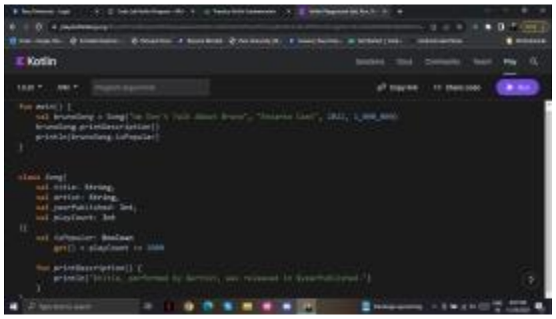
Properties for the title, artist, year published, and play count
A property that indicates whether the song is popular. If the play count is less than 1,000, consider it unpopular.
A method that prints a song description in this format:
"[Title], performed by [artist], was released in [year published]."

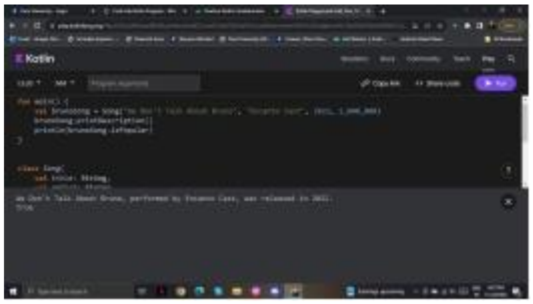


```
We Don't Talk About Bruno, performed by Encanto Cast, was released in 2022.
true
```

ANSWER:
Link: https://pl.kotl.in/v0SnpuPD_

```
fun main() {
    val brunoSong = Song("We Don't Talk About Bruno", "Encanto Cast", 2022, 1_000_000)
    brunoSong.printDescription()
    println(brunoSong.isPopular)
}



class Song(
    val title: String,
    val artist: String,
    val yearPublished: Int,
    val playCount: Int
){
    val isPopular: Boolean
        get() = playCount >= 1000

    fun printDescription() {
        println("$title, performed by $artist, was released in $yearPublished.")
    }
}
```
-The solution contains a Song class with a default constructor that accepts all required parameters. The Song class also has an isPopular property that uses a custom getter function, and a method that prints the description of itself. You can create an instance of the class in the main() function and call its methods to test whether the implementation is correct. You can use underscores when writing large numbers such as the 1_000_000 value to make it more readable.





Q5. Internet profile

Oftentimes, you're required to complete profiles for online websites that contain mandatory and non-mandatory fields. For example, you can add your personal information and link to other people who referred you to sign up for the profile.

In the initial code provided in the following code snippet, write a program which prints out a person's profile details.

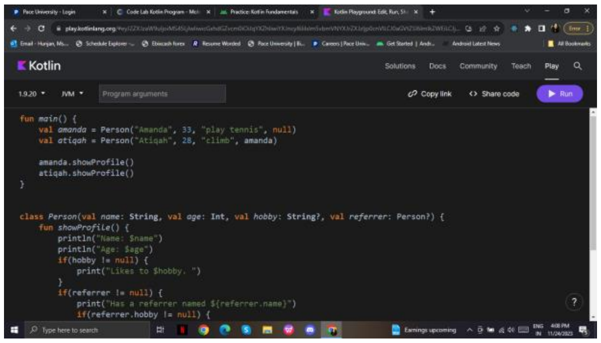
Complete the showProfile() function so that the program prints these lines:

```
Name: Amanda
Age: 33
Likes to play tennis. Doesn't have a referrer.

Name: Atiqah
Age: 28
Likes to climb. Has a referrer named Amanda, who likes to play tennis.
```

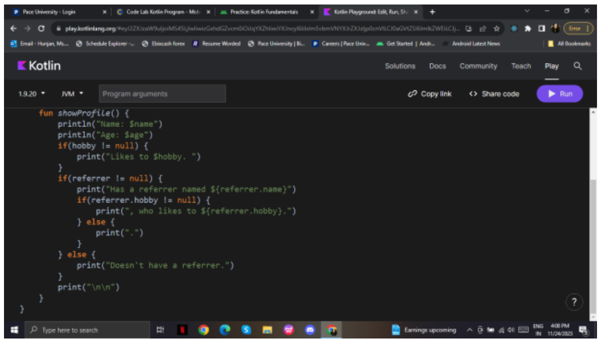
ANSWER:
Link: https://pl.kotl.in/B8-WUW2ZK

```
fun main() {
    val amanda = Person("Amanda", 33, "play tennis", null)
    val atiqah = Person("Atiqah", 28, "climb", amanda)

    amanda.showProfile()
    atiqah.showProfile()
}


class Person(val name: String, val age: Int, val hobby: String?, val referrer: Person?) {
    fun showProfile() {
        println("Name: $name")
        println("Age: $age")
        if(hobby != null) {
            print("Likes to $hobby. ")
        }
        if(referrer != null) {
            print("Has a referrer named ${referrer.name}")
            if(referrer.hobby != null) {
                print(", who likes to ${referrer.hobby}.")
            } else {
                print(".")
            }
        } else {
            print("Doesn't have a referrer.")
        }
        print("\n\n")
    }
}
```

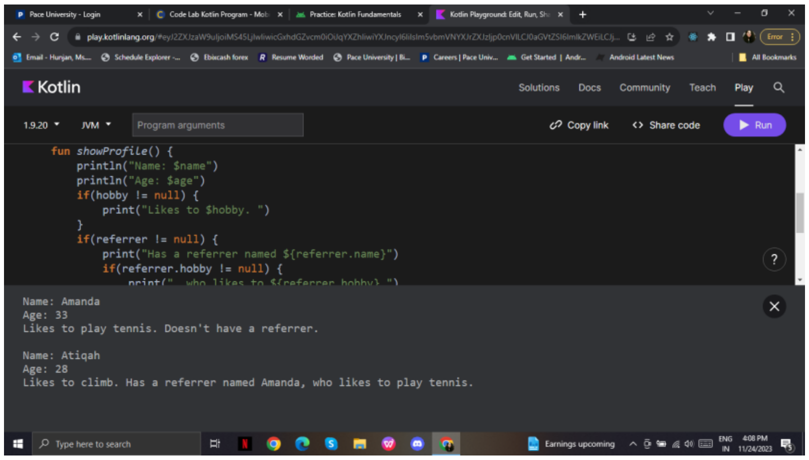
- The solution contains null checks in various if/else statements to print different text based on whether various class properties are null



```
fun main() {
    val amanda = Person("Amanda", 33, "play tennis", null)
    val atiqah = Person("Atiqah", 28, "climb", amanda)

    amanda.showProfile()
    atiqah.showProfile()
}


class Person(val name: String, val age: Int, val hobby: String?, val referrer: Person?) {
    fun showProfile() {
        println("Name: $name")
        println("Age: $age")
        if(hobby != null) {
            print("Likes to $hobby. ")
        }
        if(referrer != null) {
            print("Has a referrer named ${referrer.name}")
            if(referrer.hobby != null) {
```



```
    fun showProfile() {
        println("Name: $name")
        println("Age: $age")
        if(hobby != null) {
            print("Likes to $hobby. ")
        }
        if(referrer != null) {
            print("Has a referrer named ${referrer.name}")
            if(referrer.hobby != null) {
                print(", who likes to ${referrer.hobby}.")
            } else {
                print(".")
            }
        } else {
            print("Doesn't have a referrer.")
        }
        print("\n\n")
    }
}
```



```
    fun showProfile() {
        println("Name: $name")
        println("Age: $age")
        if(hobby != null) {
            print("Likes to $hobby. ")
        }
        if(referrer != null) {
            print("Has a referrer named ${referrer.name}")
            if(referrer.hobby != null) {
```

```
Name: Amanda
Age: 33
Likes to play tennis. Doesn't have a referrer.

Name: Atiqah
Age: 28
Likes to climb. Has a referrer named Amanda, who likes to play tennis.
```

Q6. Foldable phones

Typically, a phone screen turns on and off when the power button is pressed. In contrast, if a foldable phone is folded, the main inner screen on a foldable phone doesn't turn on when the power button is pressed.
Write a FoldablePhone class that inherits from the Phone class. It should contain the following:

- A property that indicates whether the phone is folded.

- A different switchOn() function behavior than the Phone class so that it only turns the screen on when the phone isn't folded.
- Methods to change the folding state.

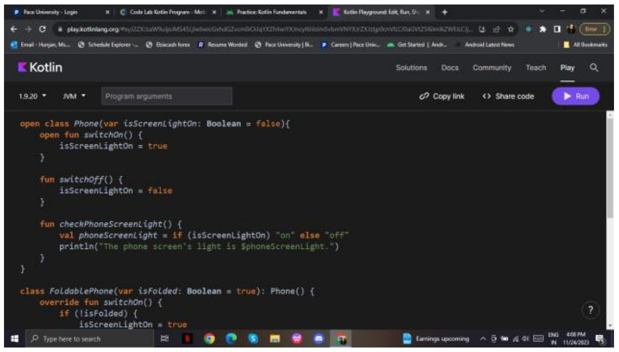
ANSWER:
Link: https://pl.kotl.in/w-kab5RKw

```
open class Phone(var isScreenLightOn: Boolean = false){
   open fun switchOn() {
      isScreenLightOn = true
   }

   fun switchOff() {
      isScreenLightOn = false
   }

   fun checkPhoneScreenLight() {
      val phoneScreenLight = if (isScreenLightOn) "on" else "off"
      println("The phone screen's light is $phoneScreenLight.")
   }
}

class FoldablePhone(var isFolded: Boolean = true): Phone() {
   override fun switchOn() {
      if (!isFolded) {
         isScreenLightOn = true
      }
   }

   fun fold() {
      isFolded = true
   }

   fun unfold() {
      isFolded = false
   }
}

fun main() {
   val newFoldablePhone = FoldablePhone()

   newFoldablePhone.switchOn()
   newFoldablePhone.checkPhoneScreenLight()
   newFoldablePhone.unfold()
   newFoldablePhone.switchOn()
   newFoldablePhone.checkPhoneScreenLight()
}
```

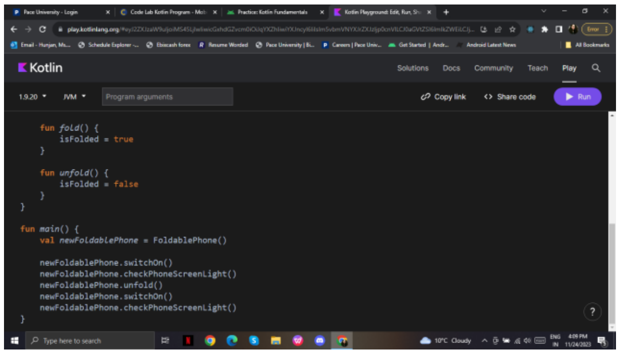
-For the Phone class to be a parent class, you need to make the class open by adding the open keyword before the class name. To override the switchOn() method in the FoldablePhone class, you need to make the method in the Phone class open by adding the open keyword before the method.

The solution contains a FoldablePhone class with a default constructor that contains a default argument for the isFolded parameter. The FoldablePhone class also has two methods *to change the isFolded property to either a true or false value. It also overrides the switchOn() method inherited from the Phone class.*

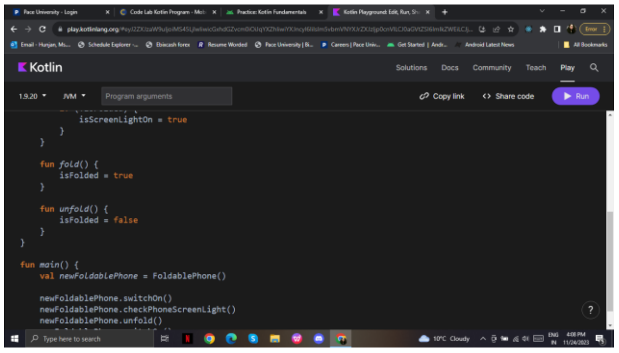
You can create an instance of the class in the main() function and call its methods to test if the implementation is correct.



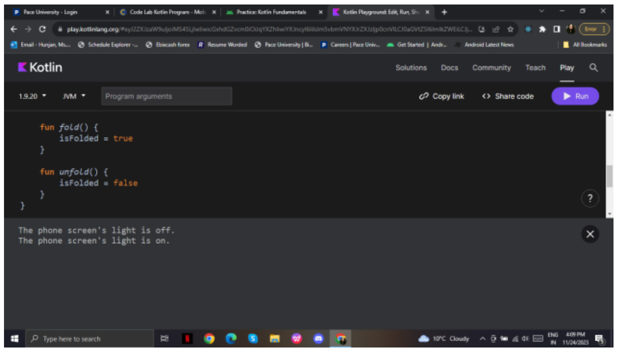
```
open class Phone(var isScreenLightOn: Boolean = false){
    open fun switchOn() {
        isScreenLightOn = true
    }

    fun switchOff() {
        isScreenLightOn = false
    }

    fun checkPhoneScreenLight() {
        val phoneScreenLight = if (isScreenLightOn) "on" else "off"
        println("The phone screen's light is $phoneScreenLight.")
    }
}

class FoldablePhone(var isFolded: Boolean = true): Phone() {
    override fun switchOn() {
        if (!isFolded) {
            isScreenLightOn = true
```



```
    fun fold() {
        isFolded = true
    }

    fun unfold() {
        isFolded = false
    }
}

fun main() {
    val newFoldablePhone = FoldablePhone()

    newFoldablePhone.switchOn()
    newFoldablePhone.checkPhoneScreenLight()
    newFoldablePhone.unfold()
    newFoldablePhone.switchOn()
    newFoldablePhone.checkPhoneScreenLight()
}
```



```
            isScreenLightOn = true
        }
    }

    fun fold() {
        isFolded = true
    }

    fun unfold() {
        isFolded = false
    }
}

fun main() {
    val newFoldablePhone = FoldablePhone()

    newFoldablePhone.switchOn()
    newFoldablePhone.checkPhoneScreenLight()
    newFoldablePhone.unfold()
```



```
    fun fold() {
        isFolded = true
    }

    fun unfold() {
        isFolded = false
    }
}
```

```
The phone screen's light is off.
The phone screen's light is on.
```

Q7. Special Auction

Typically in an auction, the highest bidder determines the price of an item. In this special auction, if there's no bidder for an item, the item is automatically sold to the auction house at the minimum price.
In the initial code provided in the following code snippet, you're given an auctionPrice() function that accepts a nullable Bid? type as an argument.
Complete the auctionPrice() function so that the program prints these lines:

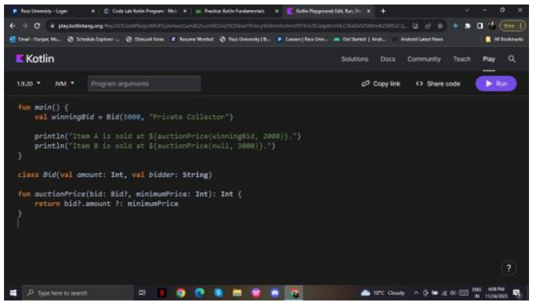
```
Item A is sold at 5000.
Item B is sold at 3000.
```

ANSWER:
Link: https://pl.kotl.in/NtOyrwLdJ

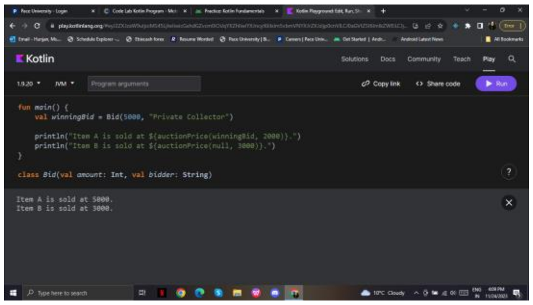
```
fun main() {
    val winningBid = Bid(5000, "Private Collector")

    println("Item A is sold at ${auctionPrice(winningBid, 2000)}.")
    println("Item B is sold at ${auctionPrice(null, 3000)}.")
}
```

```
class Bid(val amount: Int, val bidder: String)

fun auctionPrice(bid: Bid?, minimumPrice: Int): Int {
    return bid?.amount ?: minimumPrice
}
```

-The solution uses the ?. safe call operator and the ?: Elvis operator to return the correct price

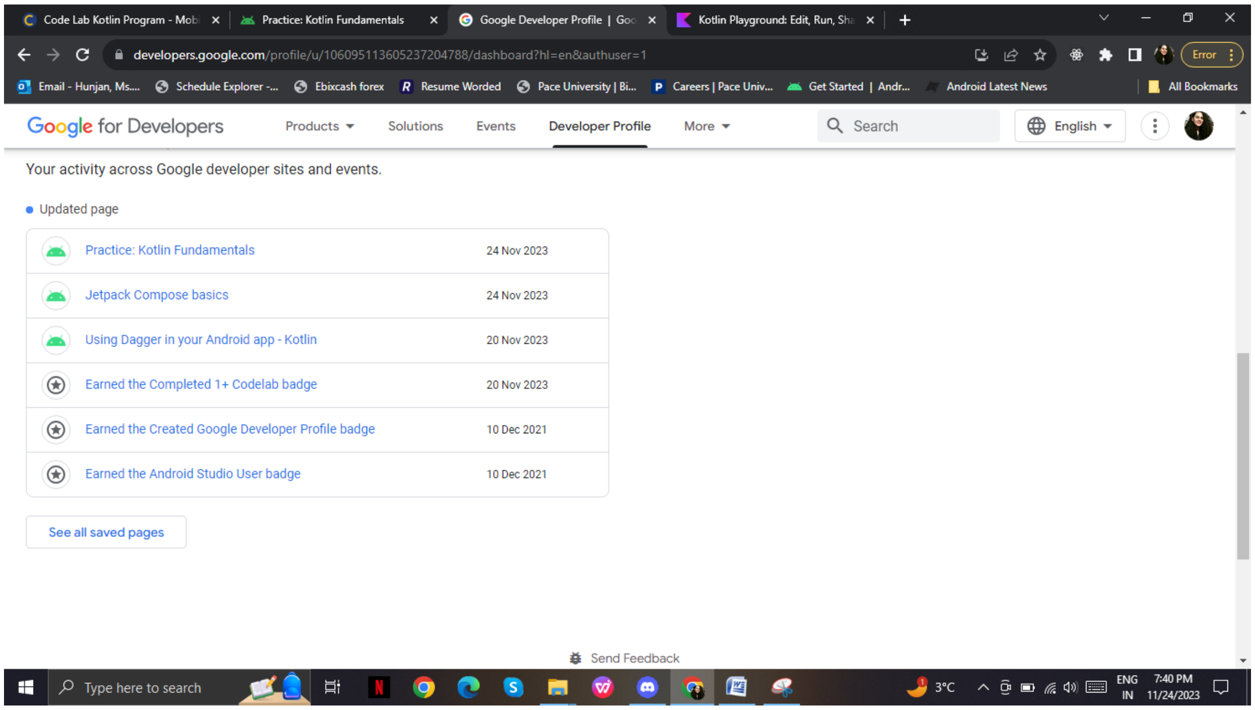


```
fun main() {
    val winningBid = Bid(5000, "Private Collector")

    println("Item A is sold at ${auctionPrice(winningBid, 2000)}.")
    println("Item B is sold at ${auctionPrice(null, 3000)}.")
}

class Bid(val amount: Int, val bidder: String)

fun auctionPrice(bid: Bid?, minimumPrice: Int): Int {
    return bid?.amount ?: minimumPrice
}
```



```
fun main() {
    val winningBid = Bid(5000, "Private Collector")

    println("Item A is sold at ${auctionPrice(winningBid, 2000)}.")
    println("Item B is sold at ${auctionPrice(null, 3000)}.")
}

class Bid(val amount: Int, val bidder: String)
```

```
Item A is sold at 5000.
Item B is sold at 3000.
```



developers.google.com/profile/u/106095113605237204788/dashboard?hl=en&authuser=1

Google for Developers

Your activity across Google developer sites and events.

● Updated page

Practice: Kotlin Fundamentals	24 Nov 2023
Jetpack Compose basics	24 Nov 2023
Using Dagger in your Android app - Kotlin	20 Nov 2023
Earned the Completed 1+ Codelab badge	20 Nov 2023
Earned the Created Google Developer Profile badge	10 Dec 2021
Earned the Android Studio User badge	10 Dec 2021

See all saved pages