1. Each person pick a piece of code from your own project (class, method, etc) that you think can be improved.

By working with Ruqian Yuan and Yue Sun, under their useful suggestion on refactoring this part code in secondFragment.kt file.

Before:

```
binding.buttonBlueColor.setOnClickListener {
   binding.viewModel?.setColor(Color.BLUE)
}

binding.buttonRedColor.setOnClickListener {
   binding.viewModel?.setColor(Color.RED)
}

binding.buttonBlackColor.setOnClickListener {
   binding.viewModel?.setColor(Color.BLACK)
}

binding.buttonGreenColor.setOnClickListener {
   binding.viewModel?.setColor(Color.GREEN)
}

binding.buttonYellowColor.setOnClickListener {
   binding.viewModel?.setColor(Color.YELLOW)
}

binding.buttonCyanColor.setOnClickListener {
   binding.viewModel?.setColor(Color.CYAN)
}

binding.buttonGaryColor.setOnClickListener {
   binding.viewModel?.setColor(Color.GRAY)
}
```

after:

```
fun setColor(color: Int) {
   binding.viewModel?.setColor(color)
}

binding.buttonBlueColor.setOnClickListener { setColor(Color.BLUE) }
binding.buttonRedColor.setOnClickListener { setColor(Color.RED) }
binding.buttonBlackColor.setOnClickListener { setColor(Color.BLACK) }
binding.buttonGreenColor.setOnClickListener { setColor(Color.GREEN) }
binding.buttonYellowColor.setOnClickListener { setColor(Color.YELLOW) }
binding.buttonCyanColor.setOnClickListener { setColor(Color.CYAN) }
binding.buttonGaryColor.setOnClickListener { setColor(Color.GRAY) }
```

The reason is that the original one has too many duplicates. By refactoring, finding the duplicate part, and rewriting this part into a single function, more clear for reading and understanding.

At first glance at the improvement in their code, I just found that they have a lot of files under the project package(there is no classification in there).

2. Pick a piece of functionality that both of your projects include. Discuss the similarities/differences between your implementations.

We picked the functionality of implementing drawing on the screen. For the similarities, Both of us created a view model to store and manage users' drawing data (I only use bitmap, they use path). In our project, The `BoardViewModel` class is mainly working on managing the state and logic of the drawing canvas. They also use ViewModels to manage drawing data created by users.

The biggest difference between us is storing the drawing. Our drawing image is stored in a byte array and directly rendered onto the Canvas by DAO. They also use DAO, But they store the image's path information in the database.

3. Pick part of your code that you don't think is well tested. Work with your partner to add more tests to exercise that part of your project and commit them to your repo

After discussion, we found that both of us were missing the database test. Therefore, we add the test on this function:

```
@Insert
fun insertAll(vararg items: Item)
```

Detail:
https://github.com/xxsyang/cs6018/blob/ab7d2b30ee23724d0bd7a37608d3bd21f9c9743e/cs6018_project/app/src/androidTest/java/com/example/cs6018_project/DatabaseTest.kt