1. Practice_collection
2. What is the checked exception and unchecked exception in Java, could you give one example?
3. Can there be multiple finally blocks?
4. When both catch and finally return values, what will be the final result?
5. What is **Runtime/unchecked exception**? what is Compile/Checked Exception?
6. What is the difference between **throw** and **throws**?
7. Run the below three pieces codes, Noticed the printed exceptions. why do we put the Null/Runtime exception before Exception ?

```
public class Main {
    public static void main(String[] args) {
        int a = 0;
        int b = 3
        String s = null;
        try {
            System.out.println(b / a);
            System.out.println(s.equals("aa"));
            throw new RuntimeException();
        } catch (ArithmeticException e) {
            e.printStackTrace();
        } catch (NullPointerException e) {
            e.printStackTrace();
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.getMessage();
        }

        System.out.println("End ...");
    }
}

public class Main {
    public static void main(String[] args) {
        int a = 0;
        int b = 3
        String s = null;
        try {
            // System.out.println(b / a);
            System.out.println(s.equals("aa"));
            throw new RuntimeException();
        } catch (ArithmeticException e) {
            e.printStackTrace();
```

```
        } catch (NullPointerException e) {
            e.printStackTrace();
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.getMessage();
        }

        System.out.println("End ...");
    }
}

public class Main {
    public static void main(String[] args) {
        int a = 0;
        int b = 3
        String s = null;
        try {
            // System.out.println(b / a);
            // System.out.println(s.equals("aa"));
            throw new RuntimeException();
        } catch (ArithmeticException e) {
            e.printStackTrace();
        } catch (NullPointerException e) {
            e.printStackTrace();
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.getMessage();
        }

        System.out.println("End ...");
    }
}
```

7. What is **optional**? why do you use it? write an optional example.
8. Why finally always be executed ?
9. Practice collection problems here: https://github.com/TAIsRich/chuwa-eij-tutorial/tree/main/02-java-core/src/main/java/com/chuwa/exercise/collection
10. What are the types of design patterns in Java ?
11. What are the **SOLID** Principles ?
12. How can you achieve thread-safe singleton patterns in Java ?
13. What do you understand by the Open-Closed Principle (OCP) ?

14. Liskov's substitution principle states that if class B is a subtype of class A, then object of type A may be substituted with any object of type B. What does this actually mean? (from OA) choose your answer.
    1. It mean that if the object of type A can do something, the object of type B could also be able tp perform the same thing
    2. It means that all the objects of type A could execute all the methods present in its subtype B
    3. It means if a method is present in class A, it should also be present in class B so that the object of type B could substitute object of type A.
    4. It means that for the class B to inherit class A, objects of type B and objects of type A must be same.
15. Watch the design pattern video, and type the code, submit it to **MavenProject** folder

singleton: https://www.bilibili.com/video/BV1Np4y1z7BU?p=22
Factory: https://www.bilibili.com/video/BV1Np4y1z7BU?p=35&vd_source=310561eab1216a27f7accf859bf7f6d9
Builder: https://www.bilibili.com/video/BV1Np4y1z7BU?p=50&vd_source=310561eab1216a27f7accf859bf7f6d9
Publisher_Subscriber: https://www.bilibili.com/video/BV1Np4y1z7BU?p=114&vd_source=310561eab1216a27f7accf859bf7f6d9