HW 2

1. **What is the difference between JDK, JRE, and JVM?**
    - JDK stands for Java Development Kit. It contains JRE and other development tools, compiler, etc.
    - JRE stands for Java Run Environment. It's part of JDK, and it contains JVM and other sources like libraries.
    - JVM stands for Java Virtual Machine. It allows a Java project to be run on any platform.
2. **What are the main differences between primitive types and reference types?**
    - Primitive types store actual values and live on the stack with fixed size.
    - Reference types store addresses of objects on the heap, can be null, and have variable size.
    - "==" compares values for primitives but compares references for reference types.
3. **Is Java pass-by-value or pass-by-reference? Explain.**
    - Java is pass-by-value.
    - For primitive data types, the value is passed.
    - For objects, the value passed is the reference to the object, not the object itself. So methods receive a copy of the reference, not a copy of the object.
4. **Why are Strings immutable in Java?**
    - Strings must be immutable to enable String Constant Pool. String is reference type, so if 2 strings have same value, they have same reference. If we change 1 string, the other one will be changed as well, which make String Constant Pool impossible.
    - Guarantee thread safety
    - String has usually been used to store sensitive paths, immutability makes sure no change to those paths.
5. **What is the String Constant Pool and how does it work?**
    - It's inside the heap and used to store strings.
    - Once the developer create a string, java will go through the String Constant Pool. If the string value exists, then the string will refer to the existing text instead of creating a new one.
    - This improves performance and reduces memory usage.
6. **What is the purpose of the final keyword in Java?**
    - final keywork is used to not allow changes for the variable/method/class
    - "final" used for variable is to define constants
    - "final" used for method is to prevent override
    - "final" used for class is to prevent inheritance
7. **What does the static keyword mean for variables or methods?**

static variables / methods means the variable/method is belong to the class itself instead of an object of the class. So we can access the variable/method at the class level directly without creating an object.

8. **What is a static block and when does it run?**
    - static block means a block of code with a static keywork in the front.
    - It will be run when the class is loading, and it won't be run when object of class is created. So it will run only once.
9. **Can a static method access non-static variables? Why or why not?**
    - No, static method cannot access non-static variables
    - Because static method is class level, while non-static variables are instance level. So static method is loaded when the class is loaded, but at this time there is no instance created.
    - Non-static variable can be accessed in static method only by creating an object inside this static method.
10. **Describe the JVM loading order: static block, static variables, and constructor.**
    static variables -> static block -> constructor
11. **What is the difference between a static variable and a constant defined as public static final?**
    - A static variable is class level and is mutable
    - A public static final constant is also class level, but it has been defined as constant and cannot be changed.
12. **Explain why immutable objects (like String) are thread-safe by design. Provide an example scenario.**
    - Immutable objects are thread-safe because the situation like multiple threads are trying to modify the same object at the same time won't happen. This guarantees no race conditions.
13. **What problem does making a class final solve in terms of immutability and inheritance security?**
    - A final class doesn't allow inheritance. It prevents subclasses from breaking the immutability of the parent class. So no one could create a subclass of this class and no override issues could happen.
    - Also, it prevents inheritance-based attacks where malicious subclasses override critical methods and compromise security.