This is representative of the kinds of topics and kind of questions you may be asked on the midterm.

Virtual Dispatch - Conceptual Understanding
1.) Name one reason why someone might want to use virtual dispatch.

2.) Name one reason why someone might **not** want to use virtual dispatch.

Virtual Dispatch in Java

3.) Consider the following Java code:

```java
public interface I1 {
  public void doThing();
}
public class C1 implements I1 {
  public void doThing() { System.out.println("c1"); }
}
public class C2 implements I1 {
  public void doThing() { System.out.println("c2"); }
}
public class Main {
  public void makeCall(I1 value) {
    value.doThing();
  }
  public static void main(String[] args) {
    I1 t1 = new C1();
    I1 t2 = new C2();
    makeCall(t1);
    makeCall(t2);
  }
}
```

What is the output of the `main` method above?

4.) Consider the following code snippet:

```
public class Main {
  public static void main(String[] args) {
    Operation op1 = new AddOperation();        // line 3
    Operation op2 = new SubtractOperation(); // line 4
    int res1 = op1.doOp(5, 3);                  // line 5
    int res2 = op2.doOp(5, 3);                  // line 6
    System.out.println(res1); // line 7; should print 8
    System.out.pritnln(res2); // line 8; should print 5
  }
}
```

Define any interfaces and/or classes necessary to make this snippet print 8, followed by 2.

5.) Consider the following Java code, which simulates a lock which can be either locked or unlocked. The lock is an immutable data structure, so locking or unlocking returns a new lock in an appropriate state.

```
public class Lock {
    private final boolean locked;

    public Lock(final boolean locked) {
        this.locked = locked;
    }

    public Lock unlock() {
        if (locked) {
            System.out.println("lock unlocked");
            return new Lock(false);
        } else {
            System.out.println("lock already unlocked");
            return this;
        }
    }

    public Lock lock() {
        if (!locked) {
            System.out.println("lock locked");
            return new Lock(true);
        } else {
            System.out.println("lock already locked");
            return this;
        }
    }

    public boolean isLocked() {
        return locked;
    }
}
```

Refactor this code to use virtual dispatch, instead of using if/else. As a hint, you should have a base class/interface for Lock, and subclasses for locked and unlocked locks.

Types

6.) The code below does not compile. Why?

```
public interface MyInterface {
  public void foo();
}

public class MyClass extends MyInterface {
  public void foo() {}
  public void bar() {}

  public static void main(String[] args) {
    MyInterface a = new MyClass();
    a.bar();
}
```

7.) Java supports subtyping. Write a Java code snippet that compiles and uses subtyping.

8.) Name one reason why someone might prefer static typing over dynamic typing.

9.) Name one reason why someone might prefer dynamic typing over static typing.

9.) Name one reason why someone might prefer strong typing over weak typing.

10.) Name one reason why someone might prefer weak typing over strong typing.