1 Object-Oriented Programming

Consider the following code fragment:

```java
public class Foo {
    private int x;
    public Foo(int x) {
        this.x = x;
    }
    public static void main(String[] args) {
        int x;
        for (int i = 0; i < 10; i++) {
            x = i;
        }
        Foo foo = new Foo(5);
    }
}
```

1. What is the scope of the variable defined on line 2?
   (a) Class  (b) Constructor  (c) Main  (d) For loop

2. What is the scope of the variable defined on line 3?
   (a) Class  (b) Constructor  (c) Main  (d) For loop

3. What is the scope of the variable defined on line 8?
   (a) Class  (b) Constructor  (c) Main  (d) For loop

4. What is the scope of the variable 'x' used on line 9?
   (a) Class  (b) Constructor  (c) Main  (d) For loop

5. In the constructor call on line 11, is '5' a formal parameter or actual parameter?
   (a) Formal parameter  (b) Actual parameter
2 Memory

Consider the following code fragment:

```java
public class Foo {
    public static void bar(int x) {
        x = 3;
    }
    public static void barr(int[] arr) {
        arr[0] = 3;
    }
    public static void main(String[] args) {
        int x = 5;
        bar(x);
        System.out.println(x);
        int[] arr = new int[1];
        arr[0] = 5;
        barr(arr);
        System.out.println(arr[0]);
    }
}
```

6. What is output by the println statement on line 11?
   (a) 5 (b) 3

7. What is output by the println statement on line 15?
   (a) 5 (b) 3

8. The value stored in the variable defined on line 9 (x) is:
   (a) A memory location (b) A value

9. The value stored in the variable defined on line 12 (arr) is:
   (a) A memory location (b) A value

10. A reference variable is a variable that holds a primitive value.
    (a) True (b) False
3 Inheritance

![Class Hierarchy Diagram]

Figure 1: Example of a class hierarchy.

11. Given the hierarchy shown in Figure 1, **Shape** is a subclass of **2DShape**.
   (a) True (b) False

12. Given the hierarchy shown in Figure 1, **Rectangle** is a superclass of **Square**.
   (a) True (b) False

Consider the following two Java classes:

```java
1 public class A {
2     public A() { }
3     public void m() { System.out.println("Class A"); }
4 }
5 public class B extends A {
6     public B() { }
7     public void m() { System.out.println("Class B"); }
8 }
```

Consider now the following code fragment:

```java
1 A var1 = new B();
2 var1.m();
```

13. What does this code fragment print when it is executed?
   (a) "Class A" (b) "Class B" (c) There is a compilation error (d) There is a runtime exception
Consider now the following code fragment:

1  B var2 = new A();
2  var2.m();

14. Which line has a compilation error in the above code fragment?
   (a) Line 1 (b) Line 2 (c) Both of the above (d) None of the above

Consider now the following code fragment:

1  public class TestBankAccounts {
2      public static void main(String[] args) {
3          BankAccount bacc = new BankAccount(100);
4          bacc = new SavingsAccount(100);
5          bacc = new CheckingAccount(100);
6          double x = Math.random();
7          if (x < 0.5)
8              bacc = new SavingsAccount(500);
9          else
10              bacc = new CheckingAccount(500);
11          System.out.println(bacc.toString());
12      }
13  }

15. Lines 3-5 are an example of:
   (a) Abstraction (b) Dynamic binding (c) Casting (d) Polymorphism (e) None of the above

16. Lines 6-11 are an example of:
   (a) Abstraction (b) Dynamic binding (c) Casting (d) Polymorphism (e) None of the above

17. Which class’s toString() method is called on line 11?
   (a) BankAccount (b) CheckingAccount (c) SavingsAccount (d) I don’t know

18. The super keyword:
   (a) Refers to a subclass (b) Refers to a sibling (c) Refers to the superclass
   (d) Throws an Exception

19. The extends keyword:
   (a) Is used in a classes constructor (b) Modifies the visibility of an instance variable
   (c) Appears in the line declaring the class name

20. Polymorphism can be achieved using interfaces instead of inheritance.
   (a) True (b) False
4 Stacks

21. Given the stack shown in Figure 2, how many elements are in the stack?
   (a) 0  (b) 1  (c) 2  (d) 3  (e) 4

22. Given the stack shown in Figure 2, what kind of variable do you think stack is?
   (a) Array  (b) Linked List  (c) Graph  (d) Tree

23. Given the stack shown in Figure 2, what would s.pop() return?
   (a) 0  (b) 2  (c) 3  (d) A  (e) B  (f) C

24. Given the stack shown in Figure 2, what would be the value of top after calling s.pop() 2 times?
   (a) 0  (b) 1  (c) 2  (d) 3  (e) 4

25. Given the stack shown in Figure 2 and given how we implemented the array stack in lecture, what should return/happen after we call s.pop() 4 times?
   (a) 0  (b) -1  (c) A  (d) NullPointerException  (e) EmptyCollectionException

26. Given the stack shown in Figure 2 and given how we implemented the array stack in lecture, what should return/happen after we push 3 more letters to the stack?
   (a) NullPointerException  (b) ArrayIndexOutOfBoundsException  (c) false  
   (d) expandCapacity() is called