Mid Term Exam – Fall 2020

1. (20 points) Implement the class Stack

```
class Stack {
    int top = -1;
    int[] arr;

    Stack(){
        this.arr = new int[1000];
    }
    Stack(int a){
        this.arr = new int[a];
    }

    void push(int a){
        this.top++;
        this.arr[this.top] = a;
    }
    int pop(){
        int removed = this.arr[this.top];
        this.arr[this.top] = 0;
        this.top--;

        return removed;
    }

    int getStackSize(){
        return this.top + 1;
    }
    void displayStack(){
        for(int i=0; i<=top; i++){
            System.out.print(this.arr[i] + " ");
        }
    }
}

public class TestStack_ {
    public static void main (String[] abc) {
        Stack s = new Stack();
        s.push(1); s.push(5); s.push(8); s.push(80); s.push(2);
        s.pop();
        s.pop();

        s.displayStack();
        System.out.println("\n" + "The size of the stack: " + s.getStackSize());
    }
}
```

2. (5 points) "Encapsulation keeps the data and codes safe from external inheritance." - Explain this statement with a code example.

Encapsulation process helps us to wrap the data/object state (variables) and methods (code functionality) together as a single unit. It means we can keep variables and methods in one place known as "class." . We can say Class is the base for encapsulation mechanism and with Java encapsulation features , we can hide (restrict access) to some data members in our code, which improves our application security. It is very much possible that We can create a fully encapsulated class in Java. First , we need to declare all data members of the class private. Now we can use setter and getter methods to set and get the data in it as per our requirement , that means we can restrict our class read access and write access.

Without Encapsulation:

```
class Cls{
    int a;
}

public class Encap{
    public static void main(String[] abc){
        Cls obj = new Cls();

        obj.a = "something";
    }
}
```

With Encapsulation:

```
class Cls{
    private int a;

    // setters
    void setA(int a){
        this.a = a;
    }
    // getters
    int getA(){
        return this.a;
    }
}
public class Encap{
    public static void main(String[] abc){
        Cls obj = new Cls();

        System.out.println(obj.a); // not accessable
        obj.a = 10; // not accessable

        obj.setA(10); // accessable
        obj.getA(); //accessable
    }
}
```

3. (5 points) Write a method to randomly initialize a 2D array of size NxN with the constrain that the main diagonal locations of the array are strictly zeros.

```
void random_2d(int n){
    Random r = new Random();

    int[][] array = new int[n][n];

    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            if(i==j){
                array[i][j] = 0;
            }else{
                array[i][j] = r.nextInt();
            }
        }
    }
}
```

4. (5 points) Explain the errors in the following program.

```
public class Apple extends Fruit {
    Apple a = new Apple();
}
class Fruit {
    public Fruit(String name) {
        System.out.println("Fruit's constructor is invoked");  // Implicit super constructor Fruit() is undefined for default constr
    }
}
```

Error: Implicit super constructor Fruit() is undefined for default constructor. Must define an explicit constructor. We have to declare a "Apple()" constructer which calls super() i.e Fruit() constructor with a string parameter.

5. (1+2+2 points) What does the immutable object mean? The instance of which classes are immutable? Describe the object's immutability with a suitable code example.

The immutable objects are objects whose value can not be changed after initialization.

Conditions: 1. Final class, which is declared as final so that it can't be extended. 2. All fields should be private so that direct access to the fields is blocked. 3. No Setters 4. All mutable fields should be as final so that they can not be iterated once initialized.

```
public class Immutable {
    private final String s;

    Immutable(final String s) {
        this.s = s;
    }
    public final String getName() {
        return s;
    }

    public static void main(String[] args) {
        Immutable obj = new Immutable("Core Java Training");
        System.out.println(obj.getName());
    }
}
```

6. (2+3 points) Print and explain the output of the following programs.

```
public class ClassA {
    int a;
    ClassA(){
        System.out.println(++a);
    }
    public ClassA(int b){
        System.out.println(a+=b);
    }
}
```

```
public class ClassB {
    public static void main(String[] args) {
        ClassA a = new ClassA();
        ClassA b = new ClassA(3);
    }
}
```

Output: \ 1 \ 3

Explanation: The constructor ClassA() is called during the creation of object "a". The value of a (property of class) is increased by 1 then the value of a (property of class) is printed in the console. The constructor ClassA(int b) is called during the creation of object "b". The sum of a (property of the class) and b (parameter of the constructor) is assigned to a (property of class) and the value of a (property of the class) is printed in the console.

7. (5 points) Find and explain the problems of the following code:

```
public class ClassA {
    final static int a;
    int b;
    static void f1()
    {
        b=a; // Cannot make a static reference to the non-static field b
        f2(); // Cannot make a static reference to the non-static method f2() from the type ClassA
    }
    void f2()
    {
        int c = a;
        public char c = "a"; // Duplicate local variable c. Illegal modifier for parameter c; only final is permitted
        static double v; // Illegal modifier for parameter v; only final is permitted
        a = b;
    }
}
```

1. line 6: Cannot make a static reference to the non-static field b
2. line 7: Cannot make a static reference to the non-static method f2() from the type ClassA.
3. line 12: Duplicate local variable c. Illegal modifier for parameter c; only final is permitted.
4. line 13: Illegal modifier for parameter v; only final is permitted.