Q1. [Lab06-Q2]

Refer to the subclasses of Role in Lab06Q2, each subclass object has no instance fields to keep.

Therefore we do not need to maintain multiple objects. We can thus use the Singleton Pattern for each of these subclasses so that each have only one instance.

```
public class RLeader implements Role {









    public String genTeamContactMsg(Team team) {..}
    public String getNameAndRole(Member member) {..}
}
```

```
public class RNormalMember implements Role {




    public String genTeamContactMsg(Team team) {..}
    public String getNameAndRole(Member member) {..}
}
```

```
public class Team{
    private Member[] allMembers;
    public Team(String filepathname) throws FileNotFoundException {

        ..//setup allMembers and open the file for reading
        for (int i=0; i<tot; i++) {
            String n = inFile.next();
            char roleType = inFile.next().charAt(0);

            Role r = null;

            if (roleType=='l')  r = new RLeader();

            else /*roleType=='n'*/  r = new RNormalMember();


            allMembers[i] = new Member(n, r);
        }
        inFile.close();
    }
    ..
}
```

Q2. [Lab06-Q4]

Change of the leader **- How to solve?**

Member.java : `setRole()`
Team.java : `findMember(String name)`
`changeLeader(String newLeaderName)`
Main.java : Modify `main()`

```
Please input the file pathname: m3.txt
There are 4 members in the team: Ann Daisy[Leader] Jack Joey[x]
Enter new leader: Jack
Result: Ann Daisy Jack[Leader] Joey[x]
```

```
public class Team{

    private Member[] allMembers;

    // methods: public Member getLeader() {..}
    //          etc..

    public Member findMember(String name) {









    }

    public void changeLeader(String newLeaderName) {









    }
}
```

```
public class Member
{
    private String name;
    private Role role;
    ..

    ____________________setRole(_________) {

    }
}
```

```
public static void main(String [] args)  {

    .. //Prompt for filepathname
    Team t = new Team(filepathname);
    ..

    System.out.print("Enter new leader: ");



    System.out.println("Result: " ..
}
```

Q3. [Lab06-Q5]

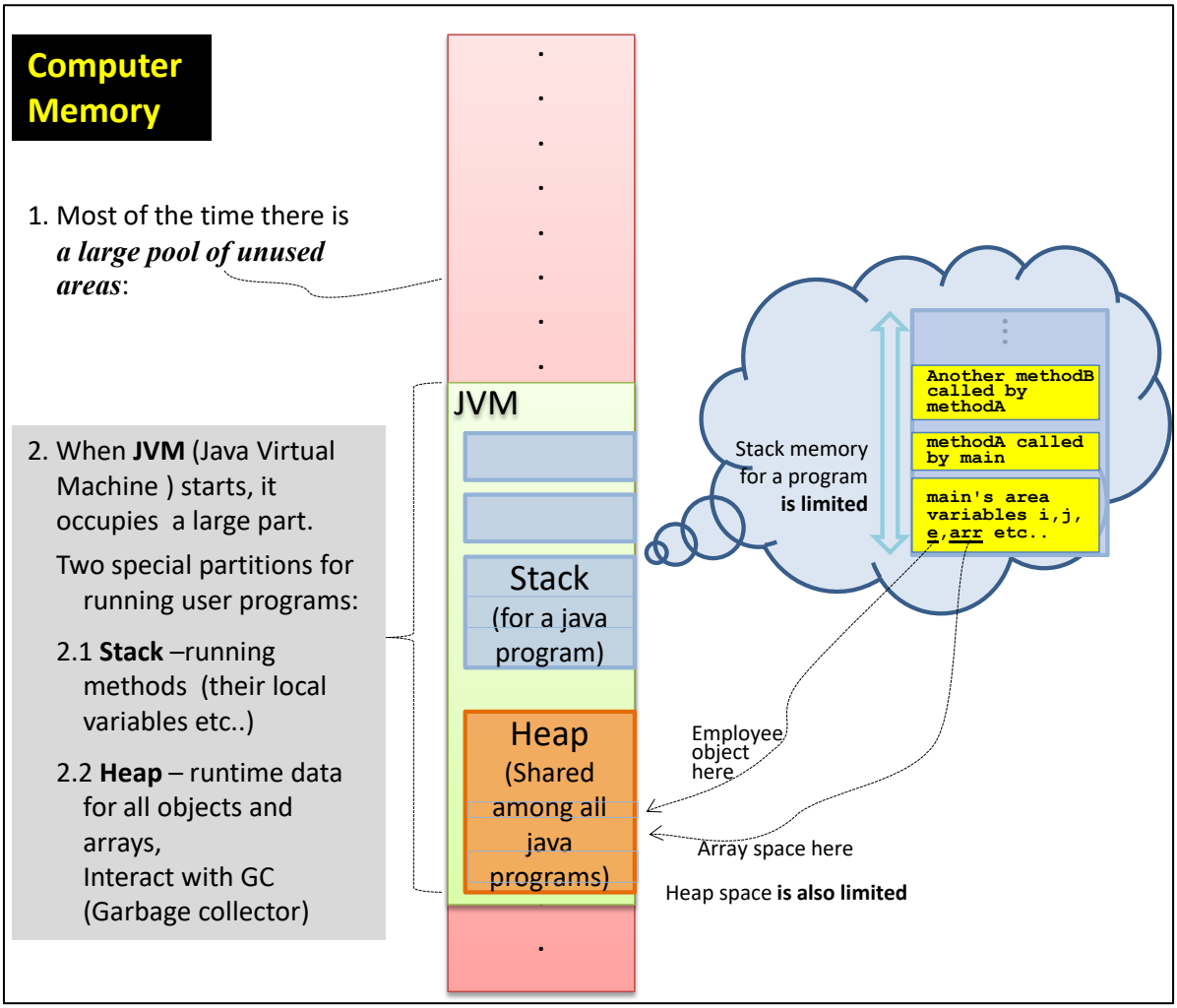
Search a member within some teams
How to read team data?

```
Please input the file pathname of each team: c:\m3.txt c:\m3a.txt c:\m3b.txt
Listing of teams:
[Team 1] 4 members: Ann Daisy[Leader] Jack Joey[x]
[Team 2] 5 members: Helena[x] Peter[x] Mary Paul Jacky[Leader]
[Team 3] 3 members: Amy Jane[Leader] Johnson[x]
Enter a name for searching: Helena
Result: Helena is a disappeared member in Team 2
```

```
public static void main(String [] args) {

    Scanner in = new Scanner(System.in);

    String filepathnames;

    System.out.print("Please input the file pathn
    filepathnames = in.nextLine();

    Scanner infnames = _____________________________

    ArrayList <Team> teams = ___________________________________

    while (infnames.hasNext()) {

        teams.add(_____________________________________);
    }

    infnames.close();

    System.out.println("\nListing of teams: ");

    int seq = 1;

    for (Team t: teams)
        System.out.printf("[Team %d] %d members: %s\n",
                seq++, t.getMemberCount(), t.getStringOfAllMembers());

    System.out.print("\nEnter a name for searching: ");
    ..
}
```

Q4. Understanding two common runtime problems: stack overflow and out of heap space

The following illustration explains the usage of memory in JVM:



When the memory usage exceeds the allocated amount, runtime errors occur:

2 kinds of runtime errors: 1. **stack overflow**: java.lang.StackOverflowError
2. **out of heap space**: java.lang.OutOfMemoryError

Your task: identify the kind of error caused by the following programs.

```
//Program 1
public static void main(String[] args)
{
    Object[] arr1 = new Object[10000000];

    Object[] arr = arr1;

    for (int i=0; i<200; i++)
    {
        arr[0] = new Object[10000000];
        arr = (Object[])arr[0];
    }
}
```

```
//Program 2
private static int factorial(int n)
{
    if (n==1)
        return 1;
    else
        return n*factorial(n+1);
}

public static void main(String[] args)
{
    int f = factorial(4);
    System.out.println(f);
}
```

Q5. Java does automatic **garbage collection**.

- If a block of memory is no longer needed, it will eventually be recycled.
- JAVA identify which objects are in use (being referenced) and which are not, and delete the unused ones.
- There are no guarantees regarding when GC will run. It is up to GC's discretion.
  https://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/garbage_collect.html

Your task:
[Refer to previous question] Explain why the error of program 1 is removed if we rewrite it as follow.

```
//Program 1
public static void main(String[] args)
{
    Object[] arr1 = new Object[10000000];
    Object[] arr = arr1;
    for (int i=0; i<200; i++)
    {
        arr[0] = new Object[10000000];
        arr = (Object[])arr[0];
    }
}
```

out of heap space:
java.lang.OutOfMemoryError

```
//Program 1 Rewritten
public static void main(String[] args)
{
    Object[] arr1 = new Object[10000000];
    Object[] arr = new Object[10000000];
    for (int i=0; i<200; i++)
    {
        arr[0] = new Object[10000000];
        arr = (Object[])arr[0];
    }
}
```

Okay