Question#1:

A. 4 6.4

B. 1 5

C. A constructed 1·25

B constructed 1·25

0
0 at least
0

1 for identifying.

0 (called recursively until stack overflow)

Question#2:

A.
```
public LogToFile() { }
```
— 2
```
public LogToFile(String fileName) {
  try {
        FileWriter fw = new FileWriter(fileName);
        BufferedWriter bw = new BufferedWriter(fw);
  } catch (IOException e) {
        e.printStackTrace();
  }
}
public void logToFile(int x, int y) {
```
— 3
```
  try {
        bw.write(x);
        bw.write(" and ");
        bw.write(y);
  } catch (IOException e) {
        e.printStackTrace();
  }
}
```

B.

```
while(i<strArr.length) {
                        try {
                                int num = Integer.parseInt(strArr[i]);
                                numCount++;
                                System.out.println(numCount);
                        } catch (Exception e) {
                        }
                        i++;
                }
```

C.

```
static void display(ArrayList<BankAccount> ba) {
                for (BankAccount a : ba) {
                        a.display();
                }
        }
```

Rough Work

QUESTION # 3

~~2231~~

(1)

```
class ComputerSystem {
private int systemID ; string brand ; bool status ;
public ComputerSystem ( ) { } .
// get /set . as needed.
public void reseve ( ){ if( ! status ) { status = true;}
                        else { sout ("Cannot reserve "); }.
public void release ( ){ status = false; }.
}
```

(2)

```
class Student {
private int studentID; string studentName;
ArrayList <Computer System> assigned Systems;
public Student ( ) { assignedSystems = new ArrayList<> ( ) ; }.
public bool asignSystem ( Computer System cs) {
 if (cs.getStatus()){assigned Systems. add ( cs ); cs.setStatus( false );
                return true ; }
 else { return false; } .
public bool unassign System ( Computer System cs ) {
if( assignedSystem. contains(cs)){assignedSystems, remove (.es ) ;
                cs: setStatus( true);
                return true;
                }
else { return false;}
```

(3)

```
class Project {
  private int ProjectID ; String ProjectTitle ; int Progress;
  ArrayList <Student> Students;
  ArrayList <Computer Systems> systems;
public Project { students = new ArrayList <> ( );
                 systems = new ArrayList <> ( );
                }
public bool Assign Systems ( Computer System cs ) {
 if (cs.getStatus ( )){ systems.add( cs ); | cs.setStatus(false);
                        return true ; }
 else { return false ; }
```

Rough Work

```
public bool AssignStudents(Students) {
        students.add(s);
        return true; }
} // class end.][Project]]
```

(d)

```
class LabManager {
    private ArrayList<ComputerSystem> SystemInventory;
    private ArrayList<Project> ProjectList;
    public LabManager() { SystemInventory = new ArrayList<>();
                          ProjectList = new ArrayList<>();
                        }
→   public addSystem(Computer.System cs) {
        SystemInventory.add(cs);
    }
→   public scheduleProject(ArrayList<ComputerSystem> systems,
                           ArrayList<Student> students) {

        Project newProj = new Project();
for (ComputerSystem cs : systems) {
                newProj.AssignSystem(cs);
            }
For (Student s : Students) {
                newProj.AssignStudents(s);
            }
        ProjectList.add(newProj);
}
                        Project P; (or iterate list of projects here)
→   public trackProjectProgress(P) {
        sout("Progress : " + P.getProgress());
        ArrayList<ComputerSystem> cList = P.getSystems();
        for (ComputerSystem cs : cList) {
                cs.display(); // or display individually
        }
```

Rough Work

→ public bool updateStatus ( ) {
// user input, new progress? // user input which project?
input → updateProgress. → project (or receive as args).
if (updateProgress < 0 || updateProgress > 100) {
// invalid try again
}
if (updateProgress != Proj.getProgress( )){
proj.setProgress (updateProgress); return true;
} else { return false; }.
}.
if (proj.getProgress( ) == 100){
} // class end [lab manger]. //release systems.
}

(e) // Main Function //
→ ComputerSystem cs1, cs2; // or you used = new computerSystem( ); = " " ( );
ArrayList<ComputerSystem> toAdd = new ArrayList( ); → constructor, parameterized
toAdd.add(c1);
toAdd.Add(c2);
→ Student s1 = new Student( );
// s2 = " " ( ); // or parameterized.
toAddStudendt.Add(s1);
// " (s2);
ArrayList<Students> toAddStudents = new ArrayList<>( );
→ Project Manager pm = new Project Manager( );

→ pm. addSystem (s1);
pm. addSystem (s2);

→ pm. ScheduleProject (toAdd, toAddStudents);

Rough Work

QUESTION # 4

(a)

```
abstract class GameObject {
    private int x; int y;
    // float speed, float gravity;
    // constructor:
    // getter setter as needed.
    void accelerate (float speed) { this.speed += speed; }
    void applyGravity (float gravity) { this.gravity += gravity; }
}
```

(b)

```
interface Movable {
    void move (int x, int y);
}
interface Attackable {
    void takeDamage (int dmg);
}
```

(c)

(1)

```
class mainCharacter extends GameObject implements
                    Movable, Attackable {
    int HP; String alignment;
```

(2)

```
    @override
    void move (int x, int y) {
        int dist = √((x - this.x)² + (y - this.y)²);
        int time = distance / speed;
        System.out.println("Character moved from " + this.x + "," + this.y + " to " + x + "," + y +
                " in " + time + " seconds");
        this.x = x;
        this.y = y;
    }
```

(a) @override

(2) void takeDamage (int damage) throws NegativeHPException {
int mod;
if (character == "good") { HP = HP - dmg*2; mod = 2;
if (character == "chaotic good" || character == "chaotic evil") {
HP = HP - dmg * 1.5; mod = 1.5;
}
if (character == "evil") { HP = HP - dmg; mod = 1; }
{
String Above
Sout ("Attack Strength" + dmg + "Modifier" + mod + "Damage Taken"
+ (dmg*mod) + "HP." + HP);
logtofile (stringAbove);
if (HP < 0)
throw new NegativeException();

(d) void logToFile (String strAbove) {
BufferedWriter bw = new BufferedWriter (new FileWriter ("Damage log.txt"
bw.write (strAbove);
}

(e) MainCharacter mc = new MainCharacter ();
mc.move (1, 1);
(2) try { mc.takeDamage (999);
} catch (NegativeHPException ex) {
// handled.
}

QUESTION #5

```
class TheatreHall {
    static int numpeople;
    void enter() {
        try {
            if (numpeople < 100)
                numpeople++;
            else
                throw Exception("Theatre full, cannot enter");
        }
        catch {
            // handle
        }
    }
    void exit() { exception handling++
        (same but on numpeople < 0)
    }
}
```

(1) (2) (3) (4)