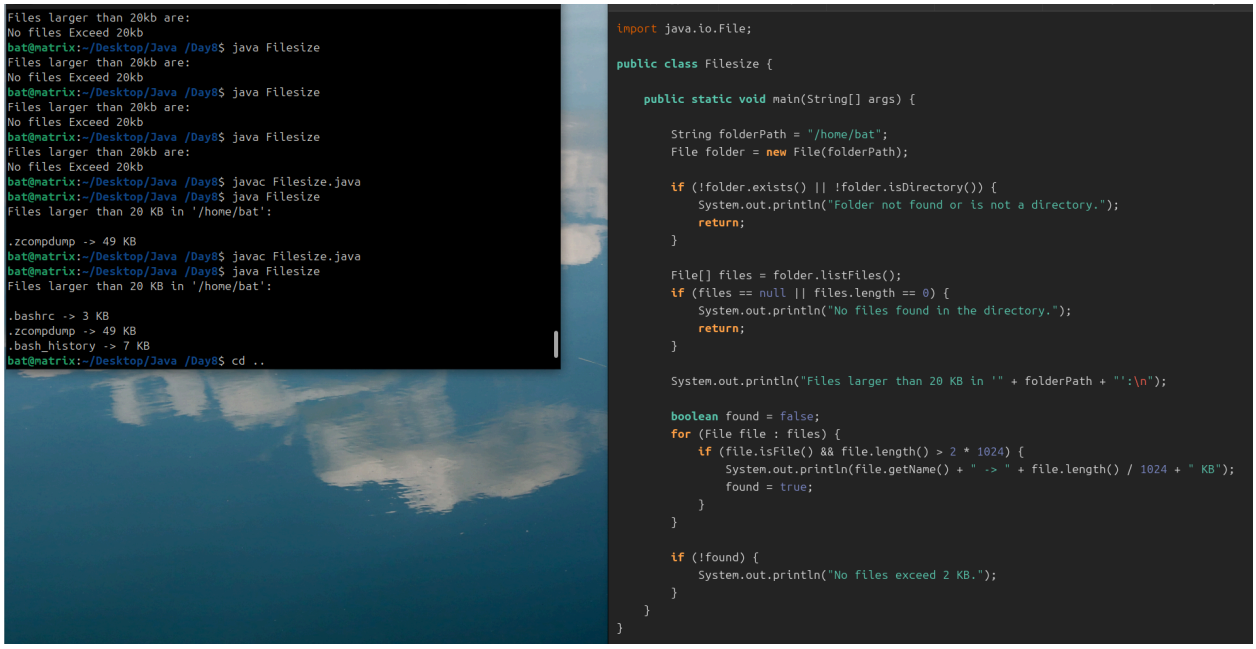
1.Java Pgm to find which files are exceeding 2Kbs of size

```
import java.io.File;

public class Filesize {

    public static void main(String[] args) {

        String folderPath = "/home/bat";
        File folder = new File(folderPath);

        if (!folder.exists() || !folder.isDirectory()) {
            System.out.println("Folder not found or is not a
directory.");
            return;
        }

        File[] files = folder.listFiles();
        if (files == null || files.length == 0) {
            System.out.println("No files found in the directory.");
            return;
        }

        System.out.println("Files larger than 20 KB in '" +
folderPath + "':\n");

        boolean found = false;
        for (File file : files) {
            if (file.isFile() && file.length() > 2 * 1024) {
                System.out.println(file.getName() + " -> " +
file.length() / 1024 + " KB");
                found = true;
            }
        }

        if (!found) {
            System.out.println("No files exceed 2 KB.");
        }
    }
}
```



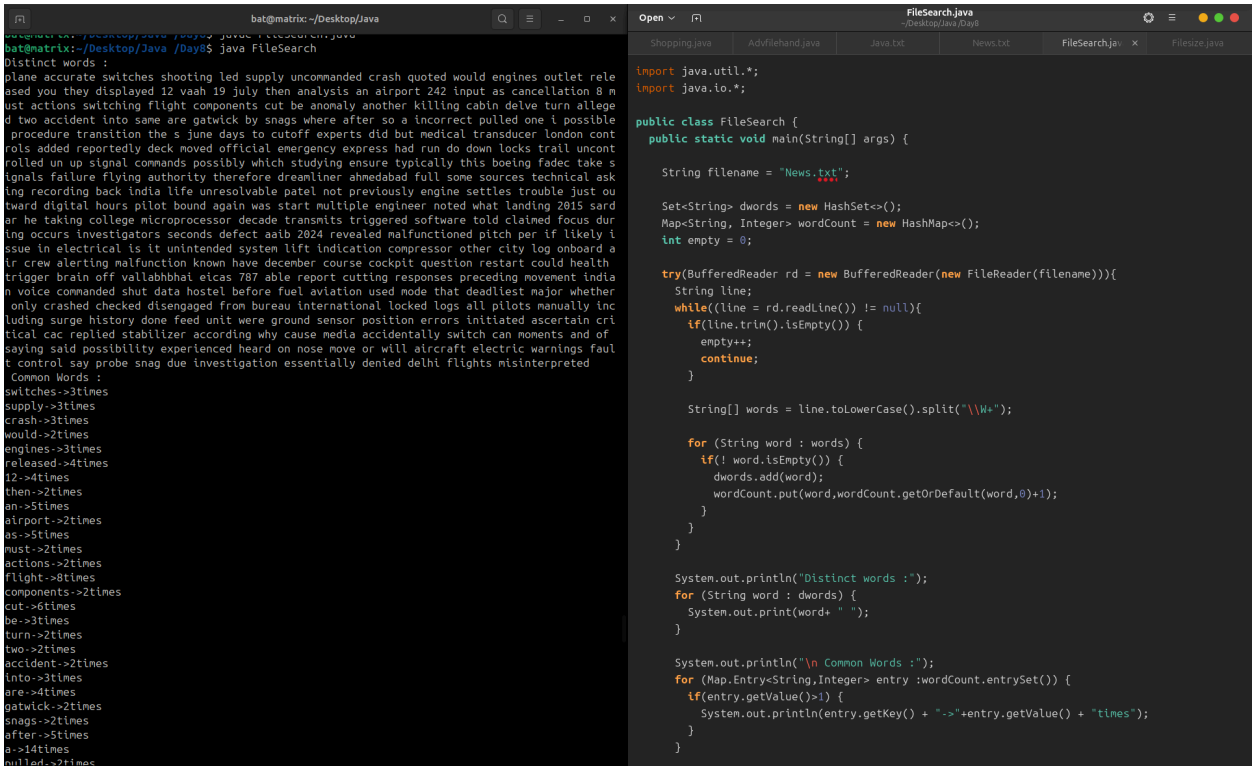
Output:

```
bat@matrix:~/Desktop/Java /Day8$ javac Filesize.java
bat@matrix:~/Desktop/Java /Day8$ java Filesize
Files larger than 20 KB in '/home/bat':

.zcompdump -> 49 KB
bat@matrix:~/Desktop/Java /Day8$ javac Filesize.java
bat@matrix:~/Desktop/Java /Day8$ java Filesize
Files larger than 20 KB in '/home/bat':

.bashrc -> 3 KB
.zcompdump -> 49 KB
.bash_history -> 7 KB
```

2.File Search - Common words,distinct words, empty lines

```
import java.util.*;
import java.io.*;

public class FileSearch {
      public static void main(String[] args) {

        String filename = "News.txt";

        Set<String> dwords = new HashSet<>();
        Map<String, Integer> wordCount = new HashMap<>();
        int empty = 0;
```

```
          try(BufferedReader rd = new BufferedReader(new
FileReader(filename))){
                String line;
                while((line = rd.readLine()) != null){
                  if(line.trim().isEmpty()) {
                    empty++;
                    continue;
                  }

                  String[] words = line.toLowerCase().split("\\W+");

                  for (String word : words) {
                    if(! word.isEmpty()) {
                      dwords.add(word);

wordCount.put(word,wordCount.getOrDefault(word,0)+1);
                    }
                  }
                }

                System.out.println("Distinct words :");
                for (String word : dwords) {
                  System.out.print(word+ " ");
                }

                System.out.println("\n Common Words :");
                for (Map.Entry<String,Integer> entry
:wordCount.entrySet()) {
                  if(entry.getValue()>1) {
                    System.out.println(entry.getKey() +
"->"+entry.getValue() + "times");
                  }
                }

                System.out.println("\nNumber of Empty Lines
:"+empty);
          } catch (IOException e) {
            System.out.println("Error reading File
:"+e.getMessage());
          }
     }
}
```



Output:

```
bat@matrix:~/Desktop/Java /Day8$ javac FileSearch.java
bat@matrix:~/Desktop/Java /Day8$ java FileSearch
Distinct words :
plane accurate switches shooting led supply uncommanded crash quoted
would engines outlet released you they displayed 12 vaah 19 july then
analysis an airport 242 input as cancellation 8 must actions
switching flight components cut be anomaly another killing cabin
delve turn alleged two accident into same are gatwick by snags where
after so a incorrect pulled one i possible procedure transition the s
june days to cutoff experts did but medical transducer london
controls added reportedly deck moved official emergency express had
run do down locks trail uncontrolled un up signal commands possibly
which studying ensure typically this boeing fadec take signals
failure flying authority therefore dreamliner ahmedabad full some
sources technical asking recording back india life unresolvable patel
not previously engine settles trouble just outward digital hours
pilot bound again was start multiple engineer noted what landing 2015
sardar he taking college microprocessor decade transmits triggered
software told claimed focus during occurs investigators seconds
defect aaib 2024 revealed malfunctioned pitch per if likely issue in
electrical is it unintended system lift indication compressor other
city log onboard air crew alerting malfunction known have december
course cockpit question restart could health trigger brain off
```

vallabhbhai eicas 787 able report cutting responses preceding movement indian voice commanded shut data hostel before fuel aviation used mode that deadliest major whether only crashed checked disengaged from bureau international locked logs all pilots manually including surge history done feed unit were ground sensor position errors initiated ascertain critical cac replied stabilizer according why cause media accidentally switch can moments and of saying said possibility experienced heard on nose move or will aircraft electric warnings fault control say probe snag due investigation essentially denied delhi flights misinterpreted

Common Words :

switches->3times
supply->3times
crash->3times
would->2times
engines->3times
released->4times
12->4times
then->2times
an->5times
airport->2times
as->5times
must->2times
actions->2times
flight->8times
components->2times
cut->6times
be->3times
turn->2times
two->2times
accident->2times
into->3times
are->4times
gatwick->2times
snags->2times
after->5times
a->14times
pulled->2times
one->2times
transition->2times
the->71times
s->3times
june->3times
to->25times
did->4times

but->3times
transducer->4times
london->3times
moved->2times
official->6times
express->3times
had->3times
down->3times
un->2times
which->2times
boeing->4times
failure->2times
dreamliner->2times
ahmedabad->5times
some->2times
technical->3times
india->3times
not->3times
engine->5times
outward->2times
pilot->5times
bound->2times
again->2times
was->4times
noted->2times
landing->2times
he->2times
taking->2times
software->2times
told->2times
claimed->2times
investigators->2times
seconds->2times
defect->2times
aaib->3times
if->3times
in->13times
electrical->2times
is->6times
it->3times
system->3times
air->4times
malfunction->5times
have->2times
could->2times

```
off->12times
787->2times
report->3times
responses->2times
indian->3times
commanded->2times
shut->2times
data->2times
fuel->9times
aviation->2times
that->8times
from->4times
including->2times
ground->2times
sensor->2times
position->7times
ascertain->3times
stabilizer->4times
according->2times
switch->3times
and->12times
of->12times
said->4times
experienced->2times
on->10times
or->5times
aircraft->9times
electric->2times
control->6times
probe->2times
due->3times
investigation->2times

Number of Empty Lines :16
```

3.Shopping Mall - implementation of Access modifiers, exceptions,inheritance

```
import java.util.*;

class Invalidexp extends Exception {
    public Invalidexp(String m) {
        super(m);
    }
}
```

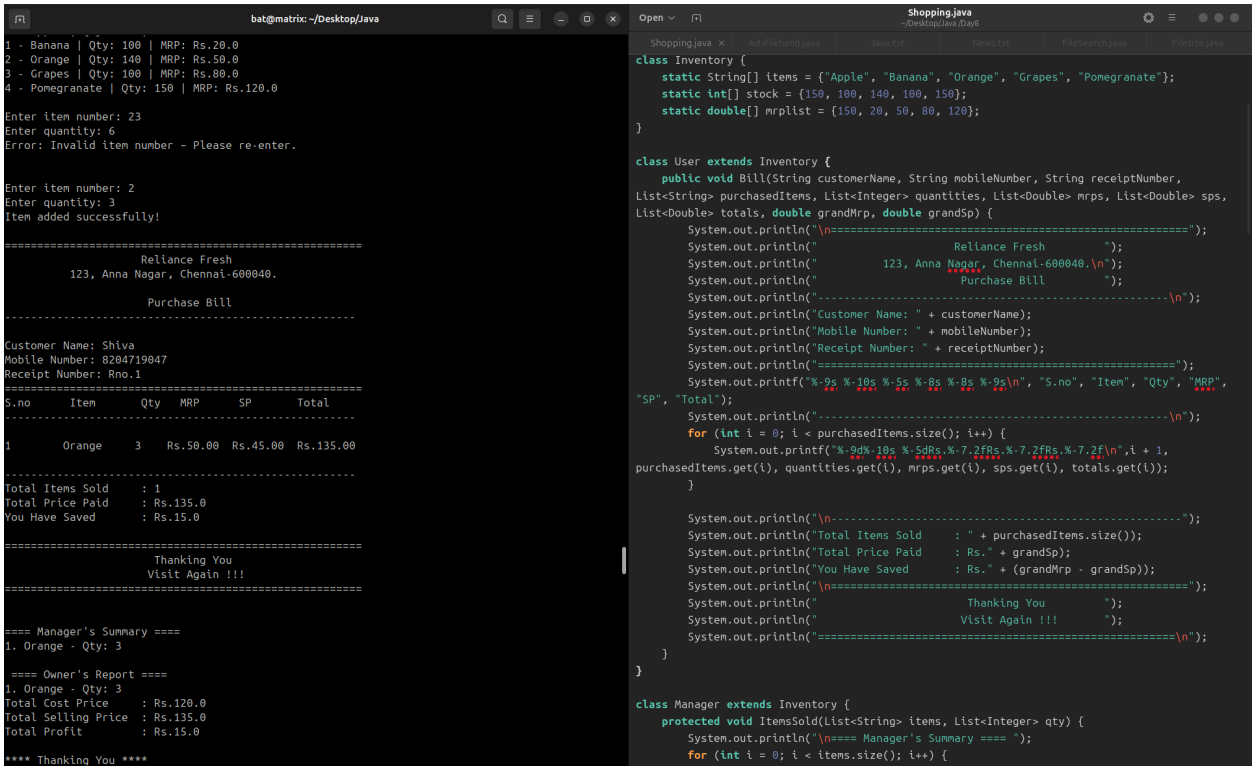
```
class Inventoryexp extends RuntimeException {
    public Inventoryexp(String m) {
        super(m);
    }
}

class Inventory {
    static String[] items = {"Apple", "Banana", "Orange", "Grapes",
"Pomegranate"};
    static int[] stock = {150, 100, 140, 100, 150};
    static double[] mrplist = {150, 20, 50, 80, 120};
}

class User extends Inventory {
    public void Bill(String customerName, String mobileNumber, String
receiptNumber, List<String> purchasedItems, List<Integer> quantities,
List<Double> mrps, List<Double> sps, List<Double> totals, double
grandMrp, double grandSp) {

System.out.println("\n==============================================
========");
        System.out.println("                      Reliance Fresh
");
        System.out.println("          123, Anna Nagar,
Chennai-600040.\n");
        System.out.println("                       Purchase Bill
");

System.out.println("----------------------------------------------
-----\n");
        System.out.println("Customer Name: " + customerName);
        System.out.println("Mobile Number: " + mobileNumber);
        System.out.println("Receipt Number: " + receiptNumber);

System.out.println("==============================================
======");
        System.out.printf("%-9s %-10s %-5s %-8s %-8s %-9s\n", "S.no",
"Item", "Qty", "MRP", "SP", "Total");

System.out.println("----------------------------------------------
-----\n");
        for (int i = 0; i < purchasedItems.size(); i++) {
            System.out.printf("%-9d%-10s
%-5dRs.%-7.2fRs.%-7.2fRs.%-7.2f\n",i + 1, purchasedItems.get(i),
quantities.get(i), mrps.get(i), sps.get(i), totals.get(i));
```

```
        }


System.out.println("\n-----------------------------------------------
-------");
        System.out.println("Total Items Sold     : " +
purchasedItems.size());
        System.out.println("Total Price Paid     : Rs." + grandSp);
        System.out.println("You Have Saved       : Rs." + (grandMrp -
grandSp));

System.out.println("\n===============================================
========");
        System.out.println("                        Thanking You
");
        System.out.println("                       Visit Again !!!
");

System.out.println("===============================================
======\n");
    }
}

class Manager extends Inventory {
    protected void ItemsSold(List<String> items, List<Integer> qty) {
        System.out.println("\n==== Manager's Summary ==== ");
        for (int i = 0; i < items.size(); i++) {
            System.out.println((i + 1) + ". " + items.get(i) + " -
Qty: " + qty.get(i));
        }
    }
}

class Owner extends Inventory {
    private void Summary(List<String> items, List<Integer> qty,
double totalcp, double totalsp) {
        System.out.println("\n ==== Owner's Report ====");
        for (int i = 0; i < items.size(); i++) {
            System.out.println((i + 1) + ". " + items.get(i) + " -
Qty: " + qty.get(i));
        }
        System.out.println("Total Cost Price     : Rs." + totalcp);
        System.out.println("Total Selling Price  : Rs." + totalsp);
        System.out.println("Total Profit         : Rs." + (totalsp -
totalcp));
```

```
    }

    public void showSummary(List<String> items, List<Integer> qty,
double totalcp, double totalsp) {
        Summary(items, qty, totalcp, totalsp);
    }
}

public class Shopping extends Inventory {
    private static int rno = 1;

    static void checkInventory(int item, int qty) {
        if (item < 0 || item >= stock.length) {
            throw new ArrayIndexOutOfBoundsException("Invalid item
number");
        }
        if (stock[item] < qty) {
            throw new Inventoryexp("Item out of stock");
        }
    }

    static void validate(int qty) throws Invalidexp {
        if (qty <= 0) throw new Invalidexp("Quantity must be greater
than 0");
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter customer name: ");
        String customerName = sc.nextLine();
        System.out.print("Enter mobile number: ");
        String mobileNumber = sc.nextLine();
        String receiptNumber = "Rno." + rno++;

        List<String> purchasedItems = new ArrayList<>();
        List<Integer> quantities = new ArrayList<>();
        List<Double> mrps = new ArrayList<>();
        List<Double> sps = new ArrayList<>();
        List<Double> cps = new ArrayList<>();
        List<Double> totals = new ArrayList<>();

        double grandMrp = 0;
        double grandSp = 0;
        double totalCp = 0;
```

```
        try {
            System.out.println("\nWelcome to Reliance Fresh");
            System.out.print("\nEnter number of different items to
purchase: ");
            int itemCount = sc.nextInt();

            while (itemCount > 0) {
                System.out.println("\n==== Available Items ====");
                for (int i = 0; i < items.length; i++) {
                    System.out.println(i + " - " + items[i] + " |
Qty: " + stock[i] + " | MRP: Rs." + mrplist[i]);
                }

                int choice = -1;
                int qty = -1;

                while (true) {
                    try {
                        System.out.print("\nEnter item number: ");
                        choice = sc.nextInt();
                        System.out.print("Enter quantity: ");
                        qty = sc.nextInt();
                        validate(qty);
                        checkInventory(choice, qty);
                        break;
                    } catch (Invalidexp | Inventoryexp |
ArrayIndexOutOfBoundsException e) {
                        System.out.println("Error: " + e.getMessage()
+ " - Please re-enter.\n");
                        sc.nextLine();
                    } catch (InputMismatchException e) {
                        System.out.println("Error: Enter valid
numeric input - Please re-enter.\n");
                        sc.nextLine();
                    }
                }

                double mrp = mrplist[choice];
                double sp = mrp * 0.9;
                double cp = mrp * 0.8;
                double total = qty * sp;

                stock[choice] -= qty;
```

```
                purchasedItems.add(items[choice]);
                quantities.add(qty);
                mrps.add(mrp);
                sps.add(sp);
                cps.add(cp);
                totals.add(total);

                grandMrp += qty * mrp;
                grandSp += total;
                totalCp += qty * cp;

                itemCount--;
                System.out.println("Item added successfully!");
            }

            User customer = new User();
            customer.Bill(customerName, mobileNumber, receiptNumber,
                    purchasedItems, quantities, mrps, sps, totals,
grandMrp, grandSp);

            Manager manager = new Manager();
            manager.ItemsSold(purchasedItems, quantities);

            Owner owner = new Owner();
            owner.showSummary(purchasedItems, quantities, totalCp,
grandSp);

        } catch (Exception e) {
            System.out.println("Unexpected Exception: " +
e.getMessage());
        } finally {
            System.out.println("\n**** Thanking You ****");
        }
    }
}
```



Output:

```
Enter customer name: Shiva
Enter mobile number: 8204719047

Welcome to Reliance Fresh

Enter number of different items to purchase: 1

==== Available Items ====
0 - Apple | Qty: 150 | MRP: Rs.150.0
1 - Banana | Qty: 100 | MRP: Rs.20.0
2 - Orange | Qty: 140 | MRP: Rs.50.0
3 - Grapes | Qty: 100 | MRP: Rs.80.0
4 - Pomegranate | Qty: 150 | MRP: Rs.120.0

Enter item number: 23
Enter quantity: 6
Error: Invalid item number - Please re-enter.


Enter item number: 2
Enter quantity: 3
Item added successfully!
```

```
=========================================================
                     Reliance Fresh
          123, Anna Nagar, Chennai-600040.

                      Purchase Bill
---------------------------------------------------------

Customer Name: Shiva
Mobile Number: 8204719047
Receipt Number: Rno.1
=========================================================
S.no      Item       Qty   MRP      SP       Total
---------------------------------------------------------

1        Orange     3    Rs.50.00  Rs.45.00  Rs.135.00

---------------------------------------------------------
Total Items Sold     : 1
Total Price Paid     : Rs.135.0
You Have Saved       : Rs.15.0

=========================================================
                       Thanking You
                      Visit Again !!!
=========================================================


==== Manager's Summary ====
1. Orange - Qty: 3

 ==== Owner's Report ====
1. Orange - Qty: 3
Total Cost Price     : Rs.120.0
Total Selling Price  : Rs.135.0
Total Profit         : Rs.15.0

**** Thanking You ****
```

4.Diamond Problem Multiple Inheritance University,Branch,Course,Dept

```
import java.util.*;

interface Univ {
    default void Show() {
```

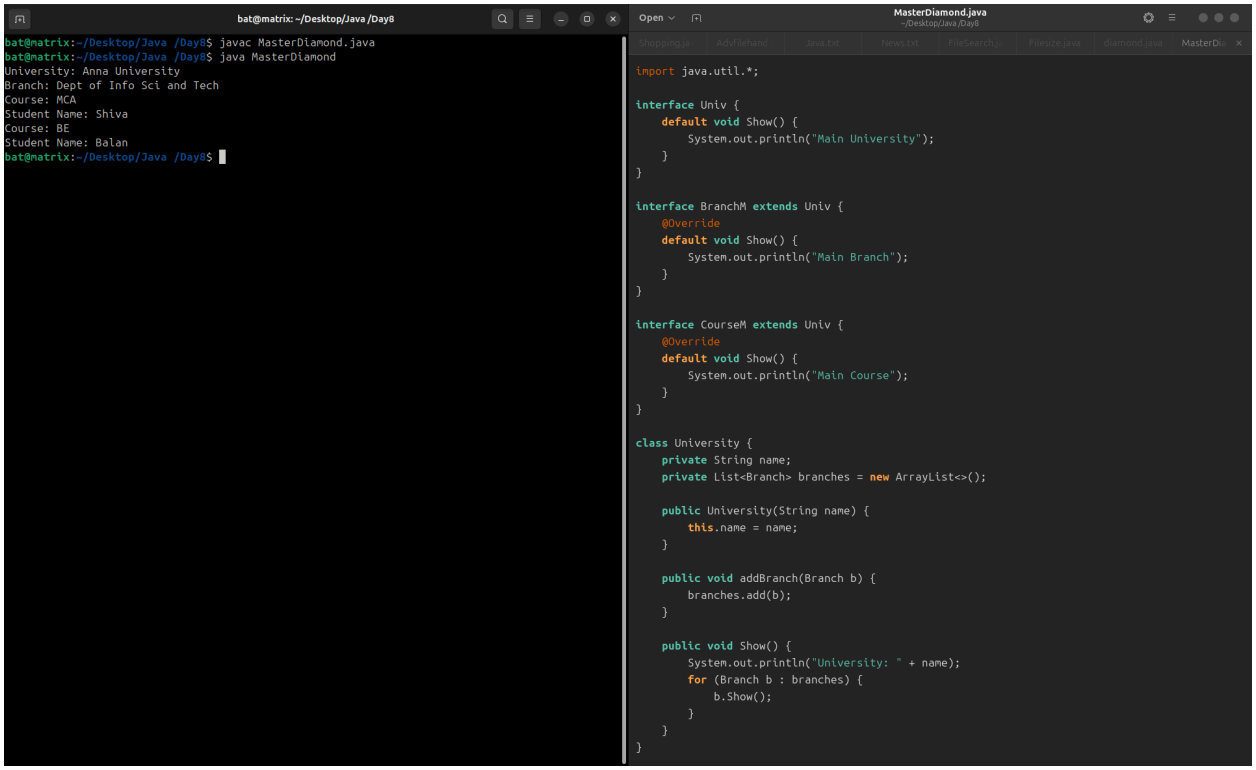
```
        System.out.println("Main University");
    }
}

interface BranchM extends Univ {
    @Override
    default void Show() {
        System.out.println("Main Branch");
    }
}

interface CourseM extends Univ {
    @Override
    default void Show() {
        System.out.println("Main Course");
    }
}

class University {
    private String name;
    private List<Branch> branches = new ArrayList<>();

    public University(String name) {
        this.name = name;
    }

    public void addBranch(Branch b) {
        branches.add(b);
    }

    public void Show() {
        System.out.println("University: " + name);
        for (Branch b : branches) {
            b.Show();
        }
    }
}

class Branch {
    private String name;
    private List<Course> courses = new ArrayList<>();

    public Branch(String name) {
        this.name = name;
    }
```

```
    public void addCourse(Course c) {
        courses.add(c);
    }

    public void Show() {
        System.out.println("Branch: " + name);
        for (Course c : courses) {
            c.Show();
        }
    }
}

class Course {
    private String name;
    private List<Student> students = new ArrayList<>();

    public Course(String name) {
        this.name = name;
    }

    public void addStudent(Student s) {
        students.add(s);
    }

    public void Show() {
        System.out.println("Course: " + name);
        for (Student s : students) {
            s.Show();
        }
    }
}

class Student implements BranchM, CourseM {
    private String name;

    public Student(String name) {
        this.name = name;
    }

    @Override
    public void Show() {
        System.out.println("Student Name: " + name);
    }
}
```

```
public class MasterDiamond {
    public static void main(String[] args) {
        University uni = new University("Anna University");

        Branch dist = new Branch("Dept of Info Sci and Tech");
        Course mca = new Course("MCA");
        Course be = new Course("BE");

        Student shiva = new Student("Shiva");
        Student balan = new Student("Balan");

        mca.addStudent(shiva);
        be.addStudent(balan);

        dist.addCourse(mca);
        dist.addCourse(be);

        uni.addBranch(dist);

        uni.Show();
    }
}
```

Output:

```
University: Anna University
Branch: Dept of Info Sci and Tech
Course: MCA
Student Name: Shiva
Course: BE
Student Name: Balan
```


bat@matrix: ~/Desktop/Java /Day8
bat@matrix:~/Desktop/Java /Day8$ javac MasterDiamond.java
bat@matrix:~/Desktop/Java /Day8$ java MasterDiamond
University: Anna University
Branch: Dept of Info Sci and Tech
Course: MCA
Student Name: Shiva
Course: BE
Student Name: Balan
bat@matrix:~/Desktop/Java /Day8$
Open
MasterDiamond.java
~/Desktop/Java /Day8
Shopping.ja
Advfilehand
Java.txt
News.txt
FileSearch.ja
Filesize.java
diamond.java
MasterDia
import java.util.*;
interface Univ {
default void Show() {
System.out.println("Main University");
}
}
interface BranchM extends Univ {
@Override
default void Show() {
System.out.println("Main Branch");
}
}
interface CourseM extends Univ {
@Override
default void Show() {
System.out.println("Main Course");
}
}
class University {
private String name;
private List<Branch> branches = new ArrayList<>();
public University(String name) {
this.name = name;
}
public void addBranch(Branch b) {
branches.add(b);
}
public void Show() {
System.out.println("University: " + name);
for (Branch b : branches) {
b.Show();
}
}
}