**AIM: Build Composite Design Pattern on the Model.**

Model chosen: Cyber Security

Tree Diagram will be like:

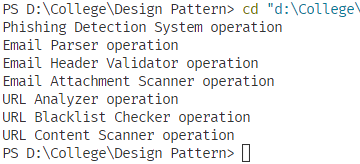
|  |
| --- |
| CyberSecurity  |  |-- PhishingDetection  | |  | |-- EmailParser  | | |  | | |-- FromAddress  | | |-- SubjectLine  | | |-- BodyContent  | |  | |-- URLAnalyzer  | | |  | | |-- URL  | | |-- Domain  | | |-- Path  | |  | |-- PhishingReport  | |  | |-- ThreatLevel  | |-- Report  | |

The tree diagram can have multiple branches, I have implemented some branches in the following code.

Code:

|  |
| --- |
| import java.util.ArrayList;  import java.util.List;  public class CompositeDP {      public static void main(String[] args) {          EmailHeaderValidator emailHeaderValidator = new EmailHeaderValidator();          EmailAttachmentScanner emailAttachmentScanner = new EmailAttachmentScanner();          URLBlacklistChecker urlBlacklistChecker = new URLBlacklistChecker();          URLContentScanner urlContentScanner = new URLContentScanner();          EmailParser emailParser = new EmailParser();          emailParser.addComponent(emailHeaderValidator);          emailParser.addComponent(emailAttachmentScanner);            URLAnalyzer urlAnalyzer = new URLAnalyzer();          urlAnalyzer.addComponent(urlBlacklistChecker);          urlAnalyzer.addComponent(urlContentScanner);          PhishingDetectionSystem phishingDetectionSystem = new PhishingDetectionSystem();          phishingDetectionSystem.addComponent(emailParser);          phishingDetectionSystem.addComponent(urlAnalyzer);          phishingDetectionSystem.operation();        }  }  abstract class Component {    protected String name;    public Component(String name) {      this.name = name;    }    public abstract void operation();  }  class Leaf extends Component {    public Leaf(String name) {      super(name);    }    @Override    public void operation() {      System.out.println(name + " operation");    }  }  class Composite extends Component {    private List<Component> components = new ArrayList<>();    public Composite(String name) {      super(name);    }    @Override    public void operation() {      System.out.println(name + " operation");      for (Component component : components) {        component.operation();      }    }    public void addComponent(Component component) {      components.add(component);    }    public void removeComponent(Component component) {      components.remove(component);    }  }  class EmailParser extends Composite {    public EmailParser() {      super("Email Parser");    }  }  class URLAnalyzer extends Composite {    public URLAnalyzer() {      super("URL Analyzer");    }  }  class EmailHeaderValidator extends Leaf {    public EmailHeaderValidator() {      super("Email Header Validator");    }  }  class EmailAttachmentScanner extends Leaf {    public EmailAttachmentScanner() {      super("Email Attachment Scanner");    }  }  class URLBlacklistChecker extends Leaf {    public URLBlacklistChecker() {      super("URL Blacklist Checker");    }  }  class URLContentScanner extends Leaf {    public URLContentScanner() {      super("URL Content Scanner");    }  }  class PhishingDetectionSystem extends Composite {    public PhishingDetectionSystem() {      super("Phishing Detection System");    }  } |

Output:



: Output of above Code