For based on the concept of "objects”,

Life, Relationships

For this project I must develop some classes.

The UML diagram of it included in the file UML. png. As we can see, there we have 10 classes: 1. Main.

java2. FileUtils. java3.

User. java4. AccessRights. java (enumeration)5. PurchaseOrder. java6. PurchaseRequisition.

java7. Item. java8. Supplier.

java9. DailyItemWiseSales. java10. TSB. javaObject-oriented programming (OOP) is a programming paradigm based on the concept of “ objects”, whichmay contain data, in the form of fields, often known as attributes; and code, in the form of procedures, oftenknown as methods. A feature of objects is that an object’s procedures can access and often modify the datafields of the object with which they are associated. In OOP, computer programs are designed by making themout of objects that interact with one another.

So, every class of my program describes some object, like User, Item, Supplier, etc. Only two classes are usedfor providing only methods. They are Main. java and FileUtils. java.

The first of them is used for interactingwith user with the help of Terminal. The second is used for writing and reading information from files. There are 4 main principles of OOP. 1.

Encapsulation: Encapsulation means that the internal representation of an object is generally hidden from view outsideof the object’s definition.

Typically, only the object’s own methods can directly inspect or manipulateits fields. Encapsulation is the hiding of data implementation by restricting access to accessors and mutators. An accessor is a method that is used to ask an object about itself.

In OOP, these are usually in the formof properties, which have a get method, which is an accessor method. However, accessor methods arenot restricted to properties and can be any public method that gives information about the state of theobject. A Mutator is public method that is used to modify the state of an object, while hiding theimplementation of exactly how the data gets modified. It’s the set method that lets the caller modifythe member data behind the scenes.

Hiding the internals of the object protects its integrity by preventing users from setting the internaldata of the component into an invalid or inconsistent state. This type of data protection andimplementation protection is called Encapsulation. A benefit of encapsulation is that it can reduce system complexity. Java provides a number of access modifiers to set access levels for classes, variables, methods, andconstructors. The four access levels are -? Visible to the package, the default. No modifiers are needed.? Visible to the class only (private).? Visible to the world (public).

? Visible to the package and all subclasses (protected). All classes of my project, that describes objects, all variables are private. So they may be accessed only withthe help of methods of this class.

So, we cannot change any field of object of classes for Item, Supplier, Purchase Requisition, Purchase Order, Daily Item-Wise Sales, or User from TBS class, which contains thisobjects. In Purchase Order class, for example, I have next private fields:/*** List of items*/private ArrayList itemCodes;/*** Quantity of items*/private HashMap quantity;/*** Date*/private String dateRequired;/*** ID of Purchase Order*/private String id;/*** Sales Manager, who adds this Purchase Order*/private User salesManager; It is possible to change their value only with the help of special method of this classes, they are called mutators(for example/*** Setter* @param code*/public void setCode(String code) {this. code = code;}method in Item class). For getting values from private variables there are special methods, which are calledaccessors (for example/*** Getter* @return*/public ArrayList getItems() {return items;}which gives access to the Items list from class, that represents Daily Item-Wise Sales). Also, there are someprivate methods, like/*** Adds users from file to list* @throws IOException*/private void addUsersFromFile() throws IOException {for (String s : FileUtils.

readFile(“ Users. txt”)) {String values = s. split(“,”); users.

add(new User(values0, values1, AccessRights. valueOf(values2)));}}in TSB. java class. So we can use this method only in this class. With the help of encapsulation, I protectobject’s variables from unauthorized changes, which may crash the program. 2. AbstractionData abstraction and encapuslation are closely tied together, because a simple definition of dataabstraction is the development of classes, objects, types in terms of their interfaces and functionality, instead of their implementation details.

Abstraction denotes a model, a view, or some other focusedrepresentation for an actual item.” An abstraction denotes the essential characteristics of an object that distinguish it from all other kindsof object and thus provide crisply defined conceptual boundaries, relative to the perspective of theviewer.” — G. BoochIn short, data abstraction is nothing more than the implementation of an object that contains the sameessential properties and actions we can find in the original object we are representing. In my project there is no abstraction, because we don’t have any standard interface of any object. 3.

InheritanceInheritance is a way to reuse code of existing objects, or to establish a subtype from an existing object, or both, depending upon programming language support. In classical inheritance where objects aredefined by classes, classes can inherit attributes and behavior from pre-existing classes called baseclasses, superclasses, parent classes or ancestor classes. The resulting classes are known as derivedclasses, subclasses or child classes. The relationships of classes through inheritance gives rise to ahierarchy.

Subclasses and Superclasses A subclass is a modular, derivative class that inherits one or moreproperties from another class (called the superclass). The properties commonly include class datavariables, properties, and methods or functions. The superclass establishes a common interface andfoundational functionality, which specialized subclasses can inherit, modify, and supplement. Thesoftware inherited by a subclass is considered reused in the subclass. In some cases, a subclass maycustomize or redefine

a method inherited from the superclass. A superclass method which can beredefined in this way is called a virtual method.

So, there is not any similar classes in my project, that is why I did not use inheritance. It is possible to createparent class like Document. java for PurchaseOrder. java and PurchaseRequisition. java classes, they havesimilar fields and methods, but this is different documents and we don’t need to incorporate them.

But allclasses in Java are inherited from Object class. So, I can say, that I use inheritance, but it is invisible. 4. PolymorphismPolymorphism means one name, many forms. Polymorphism manifests itself by having multiplemethods all with the same name, but slightly different functionality. There are 2 basic types of polymorphism.

Overridding, also called run-time polymorphism. Formethod overloading, the compiler determines which method will be executed, and this decision ismade when the code gets compiled. Overloading, which is referred to as compile-time polymorphism. Method will be used for method overriding is determined at runtime based on the dynamic type of anobject. If you can grasp these four principles, OOP can be much of a breeze for you. It might take more thanone read, I encourage you to practically try it. Polymorphism is closely connected with inheritance. We are able to Override methods of parent class.

As allclasses in Java are inherited from Object class, that is why we can override Object’s methods. In my project Ioverrides toString() method. This

method returns a string representation of the object. In general, the toStringmethod returns a string that “ textually represents” this object. The result should be a concise but informativerepresentation that is easy for a person to read. It is recommended that all subclasses override this method.

The toString method for class Object returns a string consisting of the name of the class of which the object isan instance, the at-sign character `@’, and the unsigned hexadecimal representation of the hash code of theobject. In other words, this method returns a string equal to the value of: getClass(). getName() + ‘@’ + Integer. toHexString(hashCode())But we need more convenient string representation of the object.

For example, in Purchase Order classtoString() method is override in the nextString toString() {String itemsIDs = “”; String total = “”; for (String item : itemCodes) {itemsIDs += item + “,”; total += quantity. get(item) + “,”;}itemsIDs = itemsIDs. substring(0, itemsIDs.

length() – 1); total = total. substring(0, total. length() – 1); return dateRequired + “” + purchaseManager.

getLogin() + “” + itemsIDs + “” + total;}So, the string, which represents object of this class in the next way. The first element is date, when this Orderis needed. The next, which is after symbol of new line, is login of Purchase manager, who has added this order. Then there are IDs of items, which are needed and then quantity of every item.