RONO COLINS KIPLANGAT

SCII/01363/2018

OBJECT ORIENTED IMPLEMENTATION OF REAL ESTATE USING JAVA

1. Inheritance

Class Buyer and RealEstateManager extends Person, inheriting properties and behaviours

**abstract** **class** Person {

/\* the process of reducing the object to its essence

\* so that only the necessary characteristics are exposed to the users.

\* \*/

String firstName;

String lastName;

String phoneNumber;

String location;

Person(String firstName,String lastName,String phoneNumber,String location){

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.phoneNumber = phoneNumber;

**this**.location = location;

}

//encapsulation is where a user is not allowed direct access to private data

//we use getters and setters to do this

**void** setFirstName(String fName) {

fName = firstName ;

}

String getFirstName() {

**return** firstName;

}

**void** setLastName(String lName) {

lName =lastName;

}

**void** register() {

}

}

**public** **class** Buyer **extends** Person{

**int** customerID;

Buyer(String firstName, String lastName, **int** customerID, String phoneNumber, String location) {

**super**(firstName, lastName, phoneNumber, location);

// **TODO** Auto-generated constructor stub

}

**void** inquire() {

System.***out***.println("Need more details on property 234B");

}

**void** bookProperty() {

System.***out***.println("Book property 3746B for rent");

}

**void** makeOrder() {

System.***out***.println("Confirm payment on booking");

}

}

1. Polymorphism

Person object can take forms of Buyer and RealEstateManager

**public** **class** ReManagement {

@SuppressWarnings("unused")

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

//polymorphism// person takes two forms Buyer and RealEstateManager

Person a1 = **new** Buyer(**null**, **null**, 0, **null**, **null**);

Person a2 = **new** RealEstateManager(**null**, **null**, 0, **null**, **null**);

a1.setFirstName("Collins");

System.***out***.println(a1.getFirstName());

}

}

1. Encapsulation

**abstract** **class** Person {

/\* the process of reducing the object to its essence

\* so that only the necessary characteristics are exposed to the users.

\* \*/

String firstName;

String lastName;

String phoneNumber;

String location;

Person(String firstName,String lastName,String phoneNumber,String location){

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.phoneNumber = phoneNumber;

**this**.location = location;

}

//encapsulation is where a user is not allowed direct access to private data

//we use getters and setters to do this

**void** setFirstName(String fName) {

fName = firstName ;

}

String getFirstName() {

**return** firstName;

}

**void** setLastName(String lName) {

lName =lastName;

}

**void** register() {

}

}

4.) Abstraction

**abstract** **class** Person {

/\* the process of reducing the object to its essence

\* so that only the necessary characteristics are exposed to the users.

\* \*/

String firstName;

String lastName;

String phoneNumber;

String location;

Person(String firstName,String lastName,String phoneNumber,String location){

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.phoneNumber = phoneNumber;

**this**.location = location;

}

//encapsulation is where a user is not allowed direct access to private data

//we use getters and setters to do this

**void** setFirstName(String fName) {

fName = firstName ;

}

String getFirstName() {

**return** firstName;

}

**void** setLastName(String lName) {

lName =lastName;

}

**void** register() {

}

}

5.) Interfaces

**package** softwaredev;

**public** **interface** ReamInterface {/\*

An interface in Java is a blueprint of a class

This is a blueprint of Person class

\*/

**void** setFirstName(String fName);

String getFirstName();

**void** setLastName(String lName);

**void** register();

}