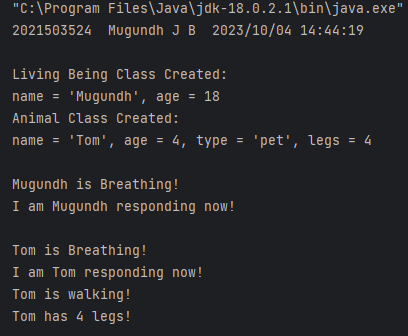
|  |
| --- |
| **Ex.No: 9 DATE: 04-10-23**  **INHERITANCE** |

1) Write a program to show single inheritance

Code:

import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
import java.util.Scanner;  
  
class LivingBeing {  
 protected String name;  
 protected int age;  
  
 // Default constructor for LivingBeing  
 public LivingBeing() {  
 this("unknown", 0);  
 }  
  
 // Parameterized constructor for LivingBeing  
 public LivingBeing(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 protected void Breath() {  
 System.*out*.println(name + " is Breathing!");  
 }  
  
 protected void Response() {  
 System.*out*.println("I am " + name + " responding now!");  
 }  
  
 @Override  
 public String toString() {  
 return "name = '" + name + '\'' +  
 ", age = " + age;  
 }  
}  
  
class Animal extends LivingBeing {  
 protected String type;  
 protected int legs;  
  
 // Default constructor for Animal  
 public Animal() {  
 this("unknown", 0);  
 }  
  
 // Parameterized constructor for Animal  
 public Animal(String type, int legs) {  
 this("unknown", 0, type, legs);  
 }  
  
 // Parameterized constructor for Animal with name, age, type, and legs  
 public Animal(String name, int age, String type, int legs) {  
 super(name, age);  
 this.type = type;  
 this.legs = legs;  
 }  
  
 public void Walk() {  
 System.*out*.println(name + " is walking!");  
 }  
  
 public void NoOfLegs() {  
 System.*out*.println(name + " has " + legs + " legs!");  
 }  
  
 @Override  
 public String toString() {  
 return super.toString() + ", " +  
 "type = '" + type + '\'' +  
 ", legs = " + legs;  
 }  
}  
  
public class SingleInheritance3524 {  
 public static void main(String[] args) {  
 // Code for getting the current date and time  
 DateTimeFormatter dtf = DateTimeFormatter.*ofPattern*("yyyy/MM/dd HH:mm:ss");  
 LocalDateTime now = LocalDateTime.*now*();  
 System.*out*.println("2021503524 " + "Mugundh J B " + dtf.format(now));  
  
 System.*out*.println();  
 // Creating instances of LivingBeing and Animal  
 LivingBeing live = new LivingBeing("Mugundh", 18);  
 Animal cat = new Animal("Tom", 4, "pet", 4);  
  
 System.*out*.println("Living Being Class: " + live);  
 System.*out*.println("Animal Class: " + cat);  
  
 System.*out*.println();  
  
 // Invoking methods on LivingBeing and Animal  
 live.Breath();  
 live.Response();  
  
 System.*out*.println();  
  
 cat.Breath();  
 cat.Response();  
 cat.Walk();  
 cat.NoOfLegs();  
 }  
}

Output:

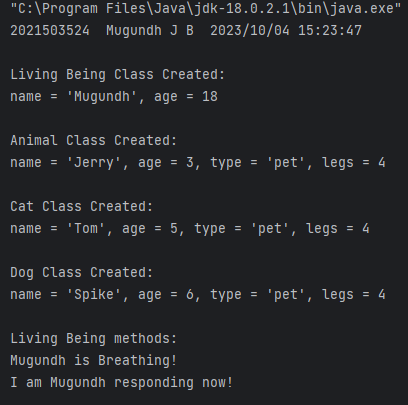


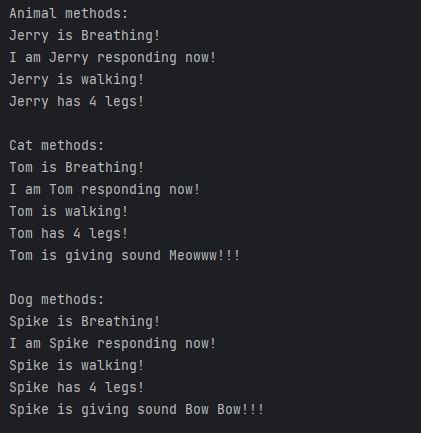
2) Write a program to show Multilevel inheritance

Code:

import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
  
class Cat extends Animal {  
 public Cat() {  
 super(); // Call the default constructor of the parent class (Animal)  
 }  
  
 public Cat(String type, int legs) {  
 super(type, legs); // Call the parameterized constructor of the parent class (Animal)  
 }  
  
 public Cat(String name, int age, String type, int legs) {  
 super(name, age, type, legs); // Call the parameterized constructor of the parent class (Animal)  
 }  
  
 // Method to simulate a cat's meow  
 public void Meow() {  
 System.*out*.println(name + " is giving sound Meowww!!!");  
 }  
  
 @Override  
 public String toString() {  
 return super.toString(); // Override the toString method   
 }  
}  
  
class Dog extends Animal {  
 public Dog() {  
 super(); // Call the default constructor of the parent class (Animal)  
 }  
  
 public Dog(String type, int legs) {  
 super(type, legs); // Call the parameterized constructor of the parent class (Animal)  
 }  
  
 public Dog(String name, int age, String type, int legs) {  
 super(name, age, type, legs); // Call the parameterized constructor of the parent class (Animal)  
 }  
  
 // Method to simulate a dog's bark  
 public void Bark() {  
 System.*out*.println(name + " is giving sound Bow Bow!!!");  
 }  
  
 @Override  
 public String toString() {  
 return super.toString(); // Override the toString method  
 }  
}  
  
public class MultilevelInheritance3524 {  
 public static void main(String[] args) {  
 // Code for getting the current date and time  
 DateTimeFormatter dtf = DateTimeFormatter.*ofPattern*("yyyy/MM/dd HH:mm:ss");  
 LocalDateTime now = LocalDateTime.*now*();  
 System.*out*.println("2021503524 " + "Mugundh J B " + dtf.format(now));  
  
 // Creating instances of LivingBeing, Animal, Cat, and Dog  
 LivingBeing live = new LivingBeing("Mugundh", 18);  
 Animal rat = new Animal("Jerry", 3, "pet", 4);  
 Cat cat = new Cat("Tom", 5, "pet", 4);  
 Dog dog = new Dog("Spike", 6, "pet", 4);  
  
 // Printing information about the created objects  
 System.*out*.println("\nLiving Being Class Created:\n" + live);  
 System.*out*.println("\nAnimal Class Created:\n" + rat);  
 System.*out*.println("\nCat Class Created:\n" + cat);  
 System.*out*.println("\nDog Class Created:\n" + dog);  
  
 // Invoking methods on Living Being  
 System.*out*.println("\nLiving Being methods: ");  
 live.Breath();  
 live.Response();  
  
 // Invoking methods on Animals  
 System.*out*.println("\nAnimal methods: ");  
 rat.Breath();  
 rat.Response();  
 rat.Walk();  
 rat.NoOfLegs();  
  
 // Invoking methods on Cat  
 System.*out*.println("\nCat methods: ");  
 cat.Breath();  
 cat.Response();  
 cat.Walk();  
 cat.NoOfLegs();  
 cat.Meow();  
  
 // Invoking methods on Dog  
 System.*out*.println("\nDog methods: ");  
 dog.Breath();  
 dog.Response();  
 dog.Walk();  
 dog.NoOfLegs();  
 dog.Bark();  
 }  
}

Output:





3) Write a program in Java to create messaging service like WhatsApp that uses single inheritance, multilevel inheritance, and hierarchical inheritance.

Code:

import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
  
// Base class User representing a user  
class User {  
 String name;  
 String phoneNumber;  
 String status;  
  
 // Constructor to initialize user attributes  
 public User(String name, String phoneNumber, String status) {  
 this.name = name;  
 this.phoneNumber = phoneNumber;  
 this.status = status;  
 }  
}  
  
// Subclass Contact, extending User to store contacts  
class Contact extends User {  
 protected User[] contacts;  
 protected int contactCount;  
 private static final int *MAX\_CONTACTS* = 10; // Default maximum contacts  
  
 // Constructor to initialize contact attributes with default maximum contacts  
 public Contact(String name, String phoneNumber, String status) {  
 super(name, phoneNumber, status);  
 contacts = new User[*MAX\_CONTACTS*];  
 contactCount = 0;  
 }  
  
 // Constructor to initialize contact attributes with a custom maximum contacts value  
 public Contact(String name, String phoneNumber, String status, int maxContacts) {  
 super(name, phoneNumber, status);  
 contacts = new User[maxContacts];  
 contactCount = 0;  
 }  
  
 // Method to add a contact to the contact list  
 public void addContact(User user) {  
 if (contactCount < contacts.length) {  
 contacts[contactCount] = user;  
 contactCount++;  
 System.*out*.println(user.name + " is added to your contacts.");  
 } else {  
 System.*out*.println("Your contact list is full.");  
 }  
 }  
}  
  
// Class Message to store message information  
class Message {  
 User sender;  
 User receiver;  
 String messageContent;  
  
 // Constructor to initialize message attributes  
 public Message(User sender, User receiver, String messageContent) {  
 this.sender = sender;  
 this.receiver = receiver;  
 this.messageContent = messageContent;  
 }  
}  
  
// Class Chat to represent a chat between two users  
class Chat {  
 private User[] participants;  
 private Message[] messages;  
 private int messageCount;  
 private static final int *MAX\_MESSAGES* = 100; // Default maximum messages  
  
 // Constructor to initialize chat participants with default maximum messages  
 public Chat(User user1, User user2) {  
 participants = new User[2];  
 participants[0] = user1;  
 participants[1] = user2;  
 messages = new Message[*MAX\_MESSAGES*];  
 messageCount = 0;  
 }  
  
 // Constructor to initialize chat participants with a custom maximum messages value  
 public Chat(User user1, User user2, int maxMessages) {  
 participants = new User[2];  
 participants[0] = user1;  
 participants[1] = user2;  
 messages = new Message[maxMessages];  
 messageCount = 0;  
 }  
  
 // Method to add a message to the chat  
 public void addMessage(Message message) {  
 if (messageCount < messages.length) {  
 messages[messageCount] = message;  
 messageCount++;  
 } else {  
 System.*out*.println("Chat history is full.");  
 }  
 }  
  
 // Method to display the chat history  
 public void displayChatHistory() {  
 System.*out*.println("\nChat History:");  
 for (Message message : messages) {  
 if (message != null) {  
 System.*out*.println(message.sender.name + " ==> " + message.receiver.name + " : " + message.messageContent);  
 }  
 }  
 }  
}  
  
public class Whatsapp3524 {  
 public static void main(String[] args) {  
 // Code for getting the current date and time  
 DateTimeFormatter dtf = DateTimeFormatter.*ofPattern*("yyyy/MM/dd HH:mm:ss");  
 LocalDateTime now = LocalDateTime.*now*();  
 System.*out*.println("2021503524 " + "Mugundh J B " + dtf.format(now));  
  
 // Create user objects  
 User alice = new User("Alice", "+1234567890", "Available");  
 User bob = new User("Bob", "+9876543210", "Away");  
  
 // Create a contact list for Alice with default maximum contacts  
 Contact aliceContacts = new Contact("Alice", "+1234567890", "Available");  
 aliceContacts.addContact(bob);  
  
 // Create messages between Alice and Bob  
 Message message1 = new Message(alice, bob, "Hi, Bob!");  
 Message message2 = new Message(bob, alice, "Hello, Alice!");  
 Message message3 = new Message(alice, bob, "...");  
 Message message4 = new Message(bob, alice, "...");  
 Message message5 = new Message(alice, bob, "Bye, Bob!");  
 Message message6 = new Message(bob, alice, "Bye, Alice!");  
  
 // Create a chat between Alice and Bob with default maximum messages  
 Chat chat = new Chat(alice, bob);  
 chat.addMessage(message1);  
 chat.addMessage(message2);  
 chat.addMessage(message3);  
 chat.addMessage(message4);  
 chat.addMessage(message5);  
 chat.addMessage(message6);  
  
 // Display the chat history  
 chat.displayChatHistory();  
 }  
}

Output:

