//Write a Java program to display the time in human readable format like,

//hours:minutes:seconds.

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

public class Time {

    public static void main(String args[]){

        DateFormat dateFormat = new SimpleDateFormat("hh:mm:ss");

        Date date = new Date();

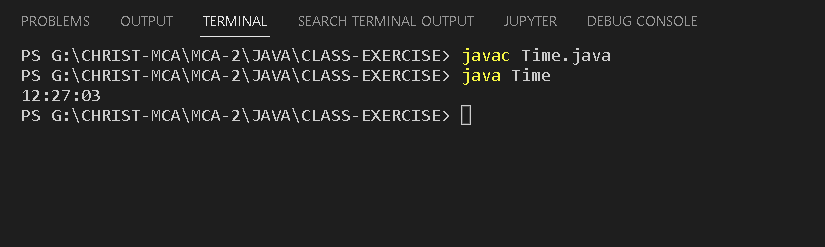
        String time = dateFormat.format(date);

        System.out.println(time);

    }

}

Output:



// Write a Java program to split a sentence in to array with the space delimiter.

// “Betty bought some butter”

public class SplitExample{

    public static void main(String args[]){

        String str = "Betty bought some butter";

        String[] newstr = str.split(" ");

        for (String a: newstr){

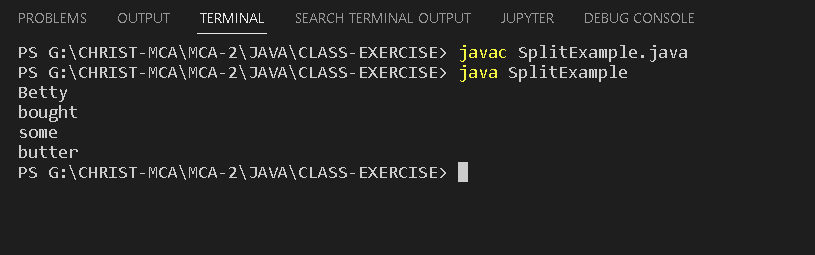
            System.out.println(a);

        }

    }

}

Output:



// Demonstrate Compile time Polymorphism and Run-time Polymorphism.

class Animal {

    // Method to make a sound

    public void makeSound() {

        System.out.println("Animal makes a sound");

    }

}

class Dog extends Animal {

    // Overriding the makeSound method in the Dog class

    @Override

    public void makeSound() {

        System.out.println("Dog barks");

    }

}

class Cat extends Animal {

    // Overriding the makeSound method in the Cat class

    @Override

    public void makeSound() {

        System.out.println("Cat meows");

    }

}

public class PolymorphismExample {

    // Method with two int parameters (compile-time polymorphism)

    public int add(int a, int b) {

        return a + b;

    }

    // Method with three int parameters (compile-time polymorphism)

    public int add(int a, int b, int c) {

        return a + b + c;

    }

    // Method with two double parameters (compile-time polymorphism)

    public double add(double a, double b) {

        return a + b;

    }

    public static void main(String[] args) {

        PolymorphismExample obj = new PolymorphismExample();

        // Demonstrating compile-time polymorphism

        System.out.println("Sum (int): " + obj.add(10, 20));

        System.out.println("Sum (int): " + obj.add(10, 20, 30));

        System.out.println("Sum (double): " + obj.add(10.5, 20.5));

        // Demonstrating run-time polymorphism

        Animal animal = new Animal();

        Dog dog = new Dog();

        Cat cat = new Cat();

        animal.makeSound(); // Calls the makeSound method of the Animal class

        dog.makeSound();    // Calls the makeSound method of the Dog class

        cat.makeSound();    // Calls the makeSound method of the Cat class

        // Using a reference of the base class to refer to an object of the derived class

        Animal anotherDog = new Dog();

        anotherDog.makeSound(); // Calls the makeSound method of the Dog class (run-time polymorphism)

    }

}

Output:

