Atividade verificadora aula 6

ContaBancaria.Java

import java.util.Date;

import java.text.SimpleDateFormat;

import java.text.NumberFormat;

import java.util.Locale;

public class ContaBancaria {

private double saldo;

private final Date dataAbertura;

// Construtor sem argumentos

public ContaBancaria() {

this.saldo = 0.00;

this.dataAbertura = new Date(); // data atual

}

// Getter para saldo

public double getSaldo() {

return saldo;

}

// Setter para saldo

public void setSaldo(double saldo) {

this.saldo = saldo;

}

// Getter para dataAbertura

public Date getDataAbertura() {

return dataAbertura;

}

// Retorna a data formatada como "dd/MM/yyyy"

public String getDataAberturaFormatada() {

SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");

return formato.format(dataAbertura);

}

// Retorna o saldo formatado como moeda brasileira

public String getSaldoFormatado() {

NumberFormat formato = NumberFormat.getCurrencyInstance(new Locale("pt", "BR"));

return formato.format(saldo);

}

// Deposita valor na conta

public void depositar(double valor) {

if (valor > 0) {

saldo += valor;

}

}

// Saca valor da conta (sem permitir saldo negativo)

public boolean sacar(double valor) {

if (valor > 0 && valor <= saldo) {

saldo -= valor;

return true;

}

return false;

}

}

TesteContaBancaria.Java

public class TesteContaBancaria {

public static void main(String[] args) {

ContaBancaria conta = new ContaBancaria();

System.out.println("Data de abertura: " + conta.getDataAberturaFormatada());

System.out.println("Saldo inicial: " + conta.getSaldoFormatado());

conta.depositar(1500.50);

System.out.println("Após depósito de R$ 1500,50: " + conta.getSaldoFormatado());

boolean saque1 = conta.sacar(500);

System.out.println("Saque de R$ 500: " + (saque1 ? "Realizado com sucesso" : "Falhou"));

System.out.println("Saldo atual: " + conta.getSaldoFormatado());

boolean saque2 = conta.sacar(2000);

System.out.println("Saque de R$ 2000: " + (saque2 ? "Realizado com sucesso" : "Falhou"));

System.out.println("Saldo final: " + conta.getSaldoFormatado());

}

}

Atividade autônoma aura: Aula 6

Questão 1) D

Questão 2) B

Atividade verificadora aula 7

Classe abstrata: Veiculo.java

public abstract class Veiculo {

public abstract float acelerar(float velocidade);

public abstract void parar();

}

Classe: Bicicleta.java

public class Bicicleta extends Veiculo {

@Override

public float acelerar(float velocidade) {

System.out.println("A bicicleta está acelerando a " + velocidade + " km/h.");

return velocidade;

}

@Override

public void parar() {

System.out.println("A bicicleta está parando.");

}

}

Classe: Automovel.java

public class Automovel extends Veiculo {

@Override

public float acelerar(float velocidade) {

System.out.println("O automóvel está acelerando a " + velocidade + " km/h.");

return velocidade;

}

@Override

public void parar() {

System.out.println("O automóvel está parando.");

}

public void mudarOleo(float litros) {

System.out.println("Trocando " + litros + " litros de óleo do automóvel.");

}

}

Classe de teste: TesteVeiculo.java

public class TesteVeiculo {

public static void main(String[] args) {

Veiculo bicicleta = new Bicicleta();

Veiculo automovel = new Automovel();

bicicleta.acelerar(15.5f);

bicicleta.parar();

// Testando o automóvel

automovel.acelerar(80.0f);

automovel.parar();

if (automovel instanceof Automovel) {

((Automovel) automovel).mudarOleo(3.5f);

}

}

}

Atividade autônoma aura: Aula 7

Questão 1) A

Questão 2) C

Atividade verificadora aula 8

Cadastro.java

import java.util.List;

public interface CadastroDAO {

void salvar(Object obj);

void atualizar(Object obj);

void deletar(int id);

Object getObjetoPorId(int id);

List<Object> listarObjetos();

}

Atividade autônoma aura: Aula 8

Questão 1) C

Questão 2) E