B nReview of Java programming basics

programs

* Hello world program.

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

* Print numbers from 1-10

public class PrintNumbers {

public static void main(String[] args) {

for (int i = 1; i <= 10; i++) {

System.out.println(i);

}

}

}

* Print array elements

public class PrintArrayElements {

public static void main(String[] args) {

int[] numbers = {1, 2, 3, 4, 5};

for (int i = 0; i < numbers.length; i++) {

System.out.println(numbers[i]);

}

}

}

* Input array elements

import java.util.Scanner;

public class InputArrayElements {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int[] numbers = new int[5];

System.out.println("Enter five numbers:");

for (int i = 0; i < numbers.length; i++) {

numbers[i] = scanner.nextInt();

}

System.out.println("Array elements:");

for (int i = 0; i < numbers.length; i++) {

System.out.println(numbers[i]);

}

}

}

* Define method to print array elements

public class PrintArrayMethod {

public static void main(String[] args) {

int[] numbers = {1, 2, 3, 4, 5};

printArray(numbers);

}

public static void printArray(int[] arr) {

for (int i = 0; i < arr.length; i++) {

System.out.println(arr[i]);

}

}

}

* Define method to Input array elements

import java.util.Scanner;

public class InputArrayMethod {

public static void main(String[] args) {

int[] numbers = new int[5];

inputArray(numbers);

printArray(numbers);

}

public static void inputArray(int[] arr) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter five numbers:");

for (int i = 0; i < arr.length; i++) {

arr[i] = scanner.nextInt();

}

}

public static void printArray(int[] arr) {

for (int i = 0; i < arr.length; i++) {

System.out.println(arr[i]);

}

}

}

* Array of objects (Students)

public class Student {

private String name;

private int age;

public Student(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

}

public class ArrayOfObjects {

public static void main(String[] args) {

Student[] students = new Student[3];

students[0] = new Student("John", 20);

students[1] = new Student("Alice", 22);

students[2] = new Student("Bob", 21);

for (int i = 0; i < students.length; i++) {

System.out.println("Name: " + students[i].getName() + ", Age: " + students[i].getAge());

}

}

}

Homework

Create a github account and upload lecture programs to your account

|  |  |
| --- | --- |
| R-1.1 | base type from the standard input device and prints it back to the standard output device. Write a short Java method, inputAllBaseTypes, that inputs a different value of each  import java.util.Scanner;  public class Main {  public static void inputAllBaseTypes() {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter a byte value: ");  byte byteValue = scanner.nextByte();  System.out.print("Enter a short value: ");  short shortValue = scanner.nextShort();  System.out.print("Enter an int value: ");  int intValue = scanner.nextInt();  System.out.print("Enter a long value: ");  long longValue = scanner.nextLong();  System.out.print("Enter a float value: ");  float floatValue = scanner.nextFloat();  System.out.print("Enter a double value: ");  double doubleValue = scanner.nextDouble();  System.out.print("Enter a boolean value (true/false): ");  boolean booleanValue = scanner.nextBoolean();  System.out.print("Enter a char value: ");  char charValue = scanner.next().charAt(0);  System.out.println("byteValue: " + byteValue);  System.out.println("shortValue: " + shortValue);  System.out.println("intValue: " + intValue);  System.out.println("longValue: " + longValue);  System.out.println("floatValue: " + floatValue);  System.out.println("doubleValue: " + doubleValue);  System.out.println("booleanValue: " + booleanValue);  System.out.println("charValue: " + charValue);  scanner.close();  }  public static void main(String[] args) {  inputAllBaseTypes();  }  } |
| R-1.2 | Suppose that we create an array A of GameEntry objects, which has an integer scores field, and we clone A and store the result in an array B. If we then immediately set A[4].score equal to 550, what is the score value of the GameEntry object referenced by B[4]?    public class GameEntry implements Cloneable {  private int score;  public GameEntry(int score) {  this.score = score;  }  public int getScore() {  return score;  }  public void setScore(int score) {  this.score = score;  }  @Override  public GameEntry clone() throws CloneNotSupportedException {  return (GameEntry) super.clone();  }  public static void main(String[] args) {  GameEntry[] A = new GameEntry[5];  A[4] = new GameEntry(100);  // Clone array A to B  GameEntry[] B = A.clone();  // Change A[4].score to 550  A[4].setScore(550);  // The score value of B[4] is also changed  System.out.println("B[4].score: " + B[4].getScore()); // Output: B[4].score: 550  }  } |
| R-1.3 | Write a short Java method, isMultiple, that takes two long values, n and m, and returns true if and only if n is a multiple of m, that is, n = mi for some integer i.  public class Main {  public static boolean isMultiple(long n, long m) {  if (m == 0) {  return false; // Division by zero is undefined  }  return n % m == 0;  }  public static void main(String[] args) {  long n = 12;  long m = 4;  boolean result = isMultiple(n, m);  System.out.println(result); // Output: true  }  } |
| R-1.4 | Write a short Java method, isEven, that takes an int i and returns true if and only if i is even. Your method cannot use the multiplication, modulus, or division operators, however.  public class Main {  public static boolean isEven(int i) {  if ((i & 1) == 0) {  return true;  } else {  return false;  }  }  public static void main(String[] args) {  int num = 6;  boolean result = isEven(num);  System.out.println(result); // Output: true  }  } |
| R-1.5 | Write a short Java method that takes an integer n and returns the sum of all positive integers less than or equal to n.  public class Main {  public static int sumPositiveIntegers(int n) {  int sum = 0;  for (int i = 1; i <= n; i++) {  sum += i;  }  return sum;  }  public static void main(String[] args) {  int n = 5;  int result = sumPositiveIntegers(n);  System.out.println(result); // Output: 15  }  } |
| R-1.6 | Write a short Java method that takes an integer n and returns the sum of all the odd positive integers less than or equal to n.  public class Main {  public static int sumOddPositiveIntegers(int n) {  int sum = 0;  for (int i = 1; i <= n; i += 2) {  sum += i;  }  return sum;  }  public static void main(String[] args) {  int n = 10;  int result = sumOddPositiveIntegers(n);  System.out.println(result); // Output: 25  }  } |
| R-1.7 | Write a short Java method that takes an integer n and returns the sum of the squares of all positive integers less than or equal to n.  public class Main {  public static int sumSquares(int n) {  int sum = 0;  for (int i = 1; i <= n; i++) {  sum += i \* i;  }  return sum;  }  public static void main(String[] args) {  int n = 4;  int result = sumSquares(n);  System.out.println(result); // Output: 30  }  } |
| R-1.8 | Write a short Java method that counts the number of vowels in a given character string.  public class Main {  public static int countVowels(String str) {  int count = 0;  str = str.toLowerCase(); // Convert the string to lowercase for case-insensitive comparison  for (int i = 0; i < str.length(); i++) {  char ch = str.charAt(i);  if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {  count++;  }  }  return count;  }  public static void main(String[] args) {  String str = "Hello, World!";  int result = countVowels(str);  System.out.println(result); // Output: 3  }  } |
| R-1.9 | Write a short Java method that uses a StringBuilder instance to remove all the punctuation from a string s storing a sentence, for example, transforming the string "Let’s try, Mike!" to "Lets try Mike".  public class Main {  public static String removePunctuation(String s) {  StringBuilder sb = new StringBuilder();  for (int i = 0; i < s.length(); i++) {  char ch = s.charAt(i);  if (Character.isLetterOrDigit(ch) || Character.isWhitespace(ch)) {  sb.append(ch);  }  }  return sb.toString();  }  public static void main(String[] args) {  String s = "Let’s try, Mike!";  String result = removePunctuation(s);  System.out.println(result); // Output: Lets try Mike  }  } |
| R-1.10 | Write a Java class, Flower, that has three instance variables of type String, int, and float, which respectively represent the name of the flower, its number of petals, and price. Your class must include a constructor method that initializes each variable to an appropriate value, and your class should include methods for setting the value of each type, and getting the value of each type.  public class Flower {  private String name;  private int numPetals;  private float price;  public Flower(String name, int numPetals, float price) {  this.name = name;  this.numPetals = numPetals;  this.price = price;  }  public void setName(String name) {  this.name = name;  }  public void setNumPetals(int numPetals) {  this.numPetals = numPetals;  }  public void setPrice(float price) {  this.price = price;  }  public String getName() {  return name;  }  public int getNumPetals() {  return numPetals;  }  public float getPrice() {  return price;  }  public static void main(String[] args) {  Flower rose = new Flower("Rose", 12, 2.99f);  System.out.println("Name: " + rose.getName());  System.out.println("Number of Petals: " + rose.getNumPetals());  System.out.println("Price: $" + rose.getPrice());  }  } |
| R-1.11 | Modify the CreditCard class from Code Fragment 1.5 to include a method that updates the credit limit.  public class CreditCard {  private String customer;  private String bank;  private String account;  private int limit;  protected double balance;  public CreditCard(String customer, String bank, String account, int limit, double balance) {  this.customer = customer;  this.bank = bank;  this.account = account;  this.limit = limit;  this.balance = balance;  }  public void updateCreditLimit(int newLimit) {  this.limit = newLimit;  }  public boolean charge(double price) {  if (price + balance > limit) {  return false;  }  balance += price;  return true;  }  public void makePayment(double amount) {  balance -= amount;  }  public String getCustomer() {  return customer;  }  public String getBank() {  return bank;  }  public String getAccount() {  return account;  }  public int getLimit() {  return limit;  }  public double getBalance() {  return balance;  }  public static void main(String[] args) {  CreditCard card = new CreditCard("John Doe", "Bank of America", "123456789", 5000, 1000);  System.out.println("Credit Card Information:");  System.out.println("Customer: " + card.getCustomer());  System.out.println("Bank: " + card.getBank());  System.out.println("Account: " + card.getAccount());  System.out.println("Limit: $" + card.getLimit());  System.out.println("Balance: $" + card.getBalance());  card.updateCreditLimit(6000);  System.out.println("Updated Credit Limit: $" + card.getLimit());  }  } |
| R-1.12 | Modify the CreditCard class from Code Fragment 1.5 so that it ignores any request to process a negative payment amount.  public class CreditCard {  private String customer;  private String bank;  private String account;  private int limit;  protected double balance;  public CreditCard(String customer, String bank, String account, int limit, double balance) {  this.customer = customer;  this.bank = bank;  this.account = account;  this.limit = limit;  this.balance = balance;  }  public void updateCreditLimit(int newLimit) {  this.limit = newLimit;  }  public boolean charge(double price) {  if (price + balance > limit) {  return false;  }  balance += price;  return true;  }  public void makePayment(double amount) {  if (amount >= 0) {  balance -= amount;  }  }  public String getCustomer() {  return customer;  }  public String getBank() {  return bank;  }  public String getAccount() {  return account;  }  public int getLimit() {  return limit;  }  public double getBalance() {  return balance;  }  public static void main(String[] args) {  CreditCard card = new CreditCard("John Doe", "Bank of America", "123456789", 5000, 1000);  System.out.println("Credit Card Information:");  System.out.println("Customer: " + card.getCustomer());  System.out.println("Bank: " + card.getBank());  System.out.println("Account: " + card.getAccount());  System.out.println("Limit: $" + card.getLimit());  System.out.println("Balance: $" + card.getBalance());  card.makePayment(200); // Positive payment amount  card.makePayment(-100); // Negative payment amount (ignored)  System.out.println("Updated Balance: $" + card.getBalance());  }  } |
| R-1.13 | Modify the declaration of the first for loop in the main method in Code Fragment 1.6 so that its charges will cause exactly one of the three credit cards to attempt to go over its credit limit. Which credit card is it?  public class TestCreditCard {  public static void main(String[] args) {  CreditCard[] wallet = new CreditCard[3];  wallet[0] = new CreditCard("John Bowman", "California Savings", "5391 0375 9387 5309", 5000);  wallet[1] = new CreditCard("John Bowman", "California Federal", "3485 0399 3395 1954", 3500);  wallet[2] = new CreditCard("John Bowman", "California Finance", "5391 0375 9387 5309", 2500);  for (int val = 1; val <= 16; val++) {  wallet[0].charge(3 \* val);  wallet[1].charge(2 \* val);  wallet[2].charge(val);  }  for (CreditCard card : wallet) {  System.out.println(card);  while (card.getBalance() > 200.0) {  card.makePayment(200);  System.out.println("New balance = " + card.getBalance());  }  }  }  } |