**Wrapper Class – Integer I**

This program is to illustrate Integer Wrapper class.

Write a program that accepts a “**Integer**” class type value as input and invokes some of the methods defined in the Integer Wrapper class.

Refer sample input and output. All functions should be performed using the methods defined in the Integer class.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output :**

Enter an integer

**540**

The binary equivalent of 540 is 1000011100

The hexadecimal equivalent of 540 is 21c

The octal equivalent of 540 is 1034

Byte value of 540 is 28

Short value of 540 is 540

Long value of 540 is 540

Float value of 540 is 540.0

Double value of 540 is 540.0

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num;

System.out.println("Enter an integer");

num = sc.nextInt();

System.out.println("The binary equivalent of "+num+" is "+ Integer.toBinaryString(num));

System.out.println("The hexadecimal equivalent of "+num+" is "+ Integer.toHexString(num));

System.out.println("The octal equivalent of "+num+" is "+ Integer.toOctalString(num));

byte b = (byte) num;

Short s = (short) num;

long l = num;

Float f = (float) num;

Double d = (double) num;

System.out.println("Byte value of "+num+" is "+b);

System.out.println("Short value of "+num+" is "+s);

System.out.println("Long value of "+num+" is "+l);

System.out.println("Float value of "+num+" is "+f);

System.out.println("Double value of "+num+" is "+d);

}

}

**Wrapper Class – 1**

Write a Java program to print the following static values defined in the Float Wrapper Class  
  
Maximum exponent a float can hold  
Maximum value a float can hold  
Number of bits used to represent a float value  
  
**Input and Output Format:**  
  
There is no input to this program.  
Refer sample output for formatting specifications.  
  
**Sample Output:**  
Maximum exponent :127  
Maximum value :3.4028235E38  
Number of bits :32

Top of Form

Bottom of Form

public class Main {

public static void main(String[] args) {

System.out.println("Maximum exponent :"+Float.MAX\_EXPONENT+" ");

System.out.println("Maximum value :"+Float.MAX\_VALUE+" ");

System.out.println("Number of bits :"+Float.SIZE );

}

}

**Converting a String to Double**

Many a times we receive the input in String and convert it into other data types (Double / Integer). Lets practice this simple exercise. Get the input amount as string and parse it using 'valueOf/parseDouble' method.

Create a main class "**Main.java**".

Create another class file "**BillHeader.java**" with the following private members

|  |  |
| --- | --- |
| **Data Type** | **Variable name** |
| Date | issueDate |
| Date | dueDate |
| Double | originalAmount |
| Double | amountOutstanding |

Include appropriate **getters**and **setters**.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

**[All text in bold corresponds to input and the rest corresponds to output]**

**Sample Input & Output:**  
Enter the issue date as dd/MM/yyyy

**12/07/2015**

Enter the due date as dd/MM/yyyy

**21/08/2015**

Enter the original amount

**2000**

Enter amount paid so far

**1000**

Issue date: 12/07/2015

Due Date: 21/08/2015

Original amount Rs.2000.0

Amount outstanding Rs.1000.0

import java.util.Date;

import java.util.Scanner;

import java.io.IOException;

import java.text.\*;

public class Main {

public static void main(String[] args) throws IOException, ParseException {

Scanner sc = new Scanner(System.in);

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

System.out.println("Enter the issue date as dd/MM/yyyy");

Date issueDate = formatter.parse(sc.nextLine());

System.out.println("Enter the due date as dd/MM/yyyy");

Date dueDate = formatter.parse(sc.nextLine());

System.out.println("Enter the original amount");

Double originalAmount = Double.parseDouble(sc.nextLine());

System.out.println("Enter amount paid so far");

Double paidAmount = Double.parseDouble(sc.nextLine());

Double amountOutstanding = originalAmount - paidAmount;

BillHeader billHeader = new BillHeader();

billHeader.setIssueDate(issueDate);

billHeader.setDueDate(dueDate);

billHeader.setOriginalAmount(originalAmount);

billHeader.setAmountOutstanding(amountOutstanding);

//System.out.println("Issue date: " + formatter.format(billHeader.getIssueDate()) + "\nDue Date: " + formatter.format(billHeader.getDueDate()) + "\nOriginal amount Rs." + billHeader.getOriginalAmount() + "\nAmount outstanding Rs." + billHeader.getAmountOutstanding());

System.out.println(billHeader);

}

}

import java.util.Date;

import java.text.\*;

public class BillHeader {

private Date issueDate;

private Date dueDate;

private Double originalAmount;

private Double amountOutstanding;

public BillHeader() {

}

public BillHeader(Date issueDate, Date dueDate, Double originalAmount, Double amountOutstanding) {

super();

this.issueDate = issueDate;

this.dueDate = dueDate;

this.originalAmount = originalAmount;

this.amountOutstanding = amountOutstanding;

}

public Date getIssueDate() {

return issueDate;

}

public void setIssueDate(Date issueDate) {

this.issueDate = issueDate;

}

public Date getDueDate() {

return dueDate;

}

public void setDueDate(Date dueDate) {

this.dueDate = dueDate;

}

public Double getOriginalAmount() {

return originalAmount;

}

public void setOriginalAmount(Double originalAmount) {

this.originalAmount = originalAmount;

}

public Double getAmountOutstanding() {

return amountOutstanding;

}

public void setAmountOutstanding(Double amountOutstanding) {

this.amountOutstanding = amountOutstanding;

}

@Override

public String toString() {

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

//return "Issue date: " + issueDate + "\nDue Date: " + dueDate + "\nOriginal amount Rs." + originalAmount + "\nAmount outstanding Rs." + amountOutstanding;

return "Issue date: " + formatter.format(getIssueDate()) + "\nDue Date: " + formatter.format(getDueDate()) + "\nOriginal amount Rs." + getOriginalAmount() + "\nAmount outstanding Rs." + getAmountOutstanding();

}

}

**String Reversal**

Write a program to reverse a given string.

**Input and Output Format:**

Input consists of a string.

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output:**

Enter a string to reverse

**Punitha**

Reverse of entered string is : ahtinuP

import java.lang.\*;

import java.io.\*;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string to reverse");

String str = sc.nextLine();

StringBuffer sbr = new StringBuffer(str);

// To reverse the string

System.out.println("Reverse of entered string is : " + sbr.reverse());

}

}

**String API : startsWith() : Illustration**

Write a program to illustrate the use of the method startsWith() defined in the string API.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output 1:**

Enter the string

**Ampphisoft**

Enter the start string

**Amphi**

"Ampphisoft" does not start with "Amphi"

**Sample Input and Output 2:**

Enter the string

**Amphisoft**

Enter the start string

**Amphi**

"Amphisoft" starts with "Amphi"

import java.util.\*;

public class Main {

public static void main(String[] args) {

// fill your code here

Scanner scan = new Scanner(System.in);

System.out.println("Enter the string");

String str = scan.nextLine();

System.out.println("Enter the start string");

// fill your code here

String chk = scan.nextLine();

if(str.startsWith(chk) )

{

System.out.println('"' + str + '"'+ " starts with "+ '"'+ chk+'"');

}

else

{

System.out.println('"' + str + '"'+ " does not start with "+ '"'+ chk+'"');

}

}

}

**String API : split() : Illustration**

This program is to illustrate the use of the method **split()** defined in the string API.

Write a program to split a string based on spaces (There may be multiple spaces too) and returns the tokens in the form of an array.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output :**

Enter the string

**Amphisoft Technologies is             a                  private     organization**

The words in the string are

Amphisoft

Technologies

is

a

private

organization

import java.util.\*;

public class Main {

public static void main(String[] args) {

// fill your code here

Scanner scan = new Scanner(System.in);

System.out.println("Enter the string");

String str = scan.nextLine();

String str2 = str.replaceAll(" +", " ");

String str1[] = str2.split(" ",-1);

System.out.println("The words in the string are");

for(int i = 0; i < str1.length; i++)

{

System.out.println(str1[i]);

}

}

}

**String Tokenizer**

Write a Java program to implement string tokenizer inorder to split a string into two different tokens by =(equal to) and ;(semicolon).  
  
**Input Format:**  
Input is a string which needs to be split.  
  
**Output Format:**  
Each line of the Output contains two strings. The first string is formed by token '=' and the second string is formed by the token ';'  
  
**Assume**: The tokens, '=' and ';', will always come alternately. Refer Sample Input.

**Sample Input:**

title=Java-Samples;author=Emiley J;publisher=java-samples.com;copyright=2007;

**Sample Output:**

title Java-Samples

author Emiley J

publisher java-samples.com

copyright 2007

Top of Form

Bottom of Form

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s = sc.nextLine();

StringTokenizer s1 = new StringTokenizer(s, "=");

String s2 = "";

while(s1.hasMoreTokens()) {

s2 = s2+" "+s1.nextToken();

}

StringTokenizer s3 = new StringTokenizer(s2, ";");

while(s3.hasMoreTokens()) {

System.out.println(s3.nextToken());

}

}

}

**Customer Address Using String Builder**

We know that Strings are immutable and are placed in a common pool called String Pool. It is always suggested that if the value of a string variable changes quite often, then the string pool would keep creating new strings and is considered as a bad practice. In these cases, the alternate way is to use StringBuilder & StringBuffer.  
  
StringBuilder would append values in a normal heap instead of any common string pools.  
The only difference between StringBuilder & StringBuffer is: StringBuilder is not Thread-Safe whereas StringBuffer is Thread-Safe.  
Let's try out using StringBuilder.  
  
Write a Java Program to display the address of the customer in a particular format.

Create a main class "**Main.java**".

Create another class file "**Address.java**" with the following private members.

|  |  |
| --- | --- |
| **Data Type** | **Variable name** |
| String | line1 |
| String | line2 |
| String | city |
| String | country |
| int | zipCode |

Include appropriate **getters**and **setters**.

Use '**toString**' method in the Address class and append using String Builder.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

**[All text in bold corresponds to input and the rest corresponds to output]**

**Sample Input & Output:**

Enter Address Details :

Enter Line 1 :

**152, South Block**

Enter Line 2 :

**Raisina Hill**

Enter City :

**New Delhi**

Enter Country :

**India**

Enter Zip Code :

**110011**

Address Details :

152, South Block,

Raisina Hill,

New Delhi - 110011

India

// fill your code here

public class Address {

private String line1;

private String line2;

private String city;

private String country;

private int zipCode;

public Address() {

super();

// TODO Auto-generated constructor stub

}

public Address(String line1, String line2, String city, String country, int zipCode) {

super();

this.line1 = line1;

this.line2 = line2;

this.city = city;

this.country = country;

this.zipCode = zipCode;

}

public String getLine1() {

return line1;

}

public void setLine1(String line1) {

this.line1 = line1;

}

public String getLine2() {

return line2;

}

public void setLine2(String line2) {

this.line2 = line2;

}

public String getCity() {

return city;

}

public void setCity(String city) {

this.city = city;

}

public String getCountry() {

return country;

}

public void setCountry(String country) {

this.country = country;

}

public int getZipCode() {

return zipCode;

}

public void setZipCode(int zipCode) {

this.zipCode = zipCode;

}

@Override

public String toString() {

return "Address Details :\n" + new StringBuilder().append(this.getLine1()) + ",\n" + new StringBuilder().append(this.getLine2()) + ",\n"

+ new StringBuilder().append(this.getCity()) + " - " + new StringBuilder().append(this.getZipCode()) + "\n"

+ new StringBuilder().append(this.getCountry());

}

}

import java.io.IOException;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException {

// fill your code here

Scanner sc = new Scanner(System.in);

System.out.println("Enter Address Details :\nEnter Line 1 :");

String line1 = sc.nextLine();

System.out.println("Enter Line 2 :");

String line2 = sc.nextLine();

System.out.println("Enter City :");

String city = sc.nextLine();

System.out.println("Enter Country :");

String country = sc.nextLine();

System.out.println("Enter Zip Code :");

int zipCode = sc.nextInt();

Address address = new Address(line1, line2, city, country, zipCode);

System.out.println(address.toString());

}

}

**String API : endsWith() : Illustration**

Write a program to illustrate the use of the method **endsWith()** defined in the string API.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output 1:**

Enter the string

**Ampphisoft**

Enter the end string

**softi**

"Ampphisoft" does not end with "softi"

**Sample Input and Output 2:**

Enter the string

**Amphisoft**

Enter the end string

**soft**

"Amphisoft" ends with "soft"

Top of Form

Bottom of Form

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scnr = new Scanner(System.in);

System.out.println("Enter the string");

String str = scnr.nextLine();

System.out.println("Enter the end string");

String strEnd = scnr.nextLine();

if( str.endsWith( strEnd ) ) {

System.out.println( "\"" + str + "\"" + " ends with \"" + strEnd + "\"" );

} else {

System.out.println( "\"" + str + "\"" + " does not end with \"" + strEnd + "\"" );

}

}

}

**Wrapper Class – Integer II**

This program is to illustrate the parseInt() method defined in the Integer Wrapper class.

Write a program that accepts 3 String values as input and invokes some of the methods defined in the Integer Wrapper class.

Refer sample input and output. All functions should be performed using the methods defined in the Integer class.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output :**

Enter the binary number

**111**

Enter the octal number

**11**

Enter the hexadecimal number

**1F**

The integer value of the binary number 111 is 7

The integer value of the octal number 11 is 9

The integer value of the hexadecimal number 1F is 31

Top of Form

Bottom of Form

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scnr = new Scanner(System.in);

System.out.println("Enter the binary number");

String binaryNo = scnr.next();

System.out.println("Enter the octal number");

String octalNo = scnr.next();

System.out.println("Enter the hexadecimal number");

String hexaDecimalNo = scnr.next();

int binNo = Integer.parseInt(binaryNo,2);

int octNo = Integer.parseInt(octalNo,8);

int hexNo = Integer.parseInt(hexaDecimalNo,16);

System.out.println( "The integer value of the binary number " + binaryNo + " is " + binNo );

System.out.println( "The integer value of the octal number " + octalNo + " is "+ octNo );

System.out.println( "The integer value of the hexadecimal number " + hexaDecimalNo + " is "+ hexNo);

}

}

**String API : replace() : Illustration**

Write a program to illustrate the use of the method **replace()** defined in the string API.

Two companies enter into a Marketing Agreement and they prepare an Agreement Draft. After that one of the companies changes its name. The name changes need to be made in the Agreement Draft as well. Write a program to perform the name changes in the agreement draft.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output :**

Enter the content of the document

**Amphi is a private organisation. Amphi is a product based company. EBox is a Amphi product**

Enter the old name of the company

**Amphi**

Enter the new name of the company

**Amphisoft**

The content of the modified document is

Amphisoft is a private organisation. Amphisoft is a product based company. EBox is a Amphisoft product

Top of Form

Bottom of Form

**Equals & == in String**

One important property of String literals are that - All String literals are placed in a common pool called String pool. If I create two string literals with the same content, both will point to the same object. Moreover, If i reassign the variable to another value, a new space is created and the existing space is left with the same value. Once a value is assigned or allocated it is never modified until its destroyed. This is why strings are called immutable.  
"==" operator compares address, but since string literals point to the same location (if values are same) it returns true.  
"equals" operator compares the content of the two strings.  
  
Lets check the above concept by validating Emailids of two customers first by using equals & equalsIgnoreCase methods.

Create a main class "**Main**.**java**".

Create another class file "**Customer**.**java**" with following private member variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| String | name |
| String | email |

Include appropriate **getters** and **setters**.  
  
Use '**equals()**' and '**equalsIgnoreCase()**' to compare the email ids[Refer sample input & output ].

**Input and Output Format:**

**Refer sample input and output for formatting specifications.**

**[All text in bold corresponds to input and the rest corresponds to output]**

**Sample Input / Output :**  
Enter First Customer Details :

Enter Customer Name :

**Roger**

Enter Customer Email :

**abc@xyz.com**

Enter Second Customer Details :

Enter Customer Name :

**Lee**

Enter Customer Email :

**abc@xYz.com**

The Email ids of Roger and Lee are not equal

Comparing without considering the cases :

The Email ids of Roger and Lee are Equal

Top of Form

Bottom of Form

**Date - 2**

Neerja Banhot was the head of the all Indian cabin crew in the Pan Am 73 flight. Neerja made sure that everything inside the flight was fine.



She noticed that the date being displayed inside the flight in an LED Board was wrong.



She asked help from a few electronics engineers who were on board. The electronics engineers figure out that the binary forms of the  date, month and year were 2 bits, 2 bits and 2 bits rotated left. Now the engineers will need to fix this. Given the incorrect date, month and the year use the Integer Wrapper Class **rotateRight()** method to print the correct date in International format.  
  
**Input Format:**  
The first line is an integer that corresponds to the incorrect day number.  
The second line is an integer that corresponds to the incorrect month number.  
The third line is an integer that corresponds to the incorrect year.  
  
**Output Format:**  
The Output should display the correct date in a single line separated by slashes in the international format.  
  
**Sample Input:**  
20  
36  
7944

**Sample Output:**  
1986/9/5

Top of Form

Bottom of Form