OOP Additional Programs

Name – Utkarsh Singh

USN – 1RV18CS181

Section – B section, 4th semester

Program 1 – Create a Java class called Student with the following details as variables within it namely USN, Name, Branch, Phone. Write a Java program to create 'n' Student objects and print the USN, Name, Branch, and Phone of these objects with suitable headings.

```
import java.util.Scanner;

class Student

{

String usn, name, branch, phone;

Student(String usn, String name, String branch, String phone)

{

this.usn = usn; this.name

= name; this.branch =

branch; this.phone =

phone;

}

public void display()

{

System.out.println(&quot;USN: &quot;+usn+&quot;\nName: &quot;+name+&quot;\nBranch:
&quot;+branch+&quot;\nPhone:&quot;+phone);

}

}

import java.util.Scanner;

public class Prog1

{

public static void main(String[] args)

{

int n;

String usn, name, branch, phone; Scanner s1 = new

Scanner(System.in); System.out.println(&quot;How many

students? &quot;); n = s1.nextInt();

Student s[] = new Student[n];
```

```
System.out.println(&quot;Enter the students&#39; details (USN, Name, Branch and Phone
Number):&quot;);

for(int i = 0; i &lt; n; i++)

{

System.out.println(&quot;Student &quot; + (i + 1));

usn = s1.next(); name

= s1.next(); branch =

s1.next(); phone =

s1.next();

s[i] = new Student(usn,name,branch,phone);

}

System.out.println(&quot;Printing students&#39; details:&quot;);

for(int i = 0; i &lt; n; i++)

{


System.out.println(&quot;Student &quot; + (i + 1));

s[i].display();

}

}

}
```

Program 2 – Write a Java program to create a base class called Bicycle with member variables (gear speed, color) of type integer and string, methods (applyBrake(int decrement), speedUp(int increment)) and constructors Bicycle(int gear, int speed, int color). Derive subclass called MountainBike from the superclass Bicycle with member variable (height) of type integer, public methods (setHeight(int newValue)) and its constructors(MountainBike(int gear,int speed,String color, int startHeight)). Create the two instances of MountainBikes and print similar Mountain bikes if the color and number of gears of mountain bikes are same. Demonstrate the code reuse and Inheritance properties of Object oriented programming by inheriting the constructors and methods of the base class.

```

package program2;

public class Bicycle {

int gear; int

speed; String

colour;

Bicycle()

{

}
```

```
Bicycle(int gear, int speed, String colour)
{
this.gear = gear;
this.speed = speed;
this.colour = colour;
}
public void applyBreak(int decrement)
{
speed -= decrement;
System.out.println(&amp;quot;Speed decreased to &amp;quot; + speed);
}
public void speedUp(int increment)
{
speed += increment;
System.out.println(&amp;quot;Speed increased to &amp;quot; + speed);
}
public boolean equals(Bicycle b)
{
if(gear == b.gear &amp;amp;&amp;amp; colour.equals(b.colour))
{
return true;
}
return false;
}
}
package program2;
public class MountainBike extends Bicycle {
int height;
MountainBike(int gear, int speed, String colour, int StartHeight)
{
super(gear,speed,colour);
height = StartHeight;

}
public void setHeight(int newValue)
{
```

```
height = newValue;
}
}
package program2;
import java.util.*;
public class Main {
public static void main(String[] args) { Scanner
s = new Scanner(System.in); int gear, speed,
height;
String colour;
MountainBike b1, b2;
System.out.println(&amp;quot;Enter the number of gears of first Mountain bike:&amp;quot;);
gear = s.nextInt();
System.out.println(&amp;quot;Enter the speed of first Mountain Bike&amp;quot;);
speed = s.nextInt();
System.out.println(&amp;quot;Enter the colour of first Mountain bike&amp;quot;);
colour = s.next();
System.out.println(&amp;quot;Enter the height of first Mountain bike&amp;quot;);
height = s.nextInt();
b1 = new MountainBike(gear, speed, colour, height);
System.out.println();
System.out.println(&amp;quot;Enter the number of gears of second Mountain bike&amp;quot;);
gear = s.nextInt();
System.out.println(&amp;quot;Enter the speed of second Mountain Bike&amp;quot;);
speed = s.nextInt();
System.out.println(&amp;quot;Enter the colour of second Mountain bike&amp;quot;);
colour = s.next();
System.out.println(&amp;quot;Enter the height of second Mountain bike&amp;quot;);
height = s.nextInt();
b2 = new MountainBike(gear, speed, colour, height);
if (b1.equals(b2)) {
System.out.println(&amp;quot;Similar Mountain bikes&amp;quot;);
} else {
System.out.println(&amp;quot;Non similar Mountain bikes&amp;quot;);
}
}     }
```

Program 3 – Write a Java program to demonstrate the following.Given a double-precision number, payment, denoting an amount of money, use the NumberFormat class' getCurrencyInstance method to convert payment into the US, Indian, Chinese, and French currency formats. Then print the formatted values as follows:

US: formattedPayment

India: formattedPayment

China: formattedPayment

France: formattedPayment

where formattedPayment is payment formatted according to the appropriate Locale's currency. Note: India does not have a built-in Locale, so you must construct one where the language is en (i.e., English).

```
package program3;

import java.util.*;

import java.text.*;

public class Main {


public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println(&quot;Enter the payment:&quot;); double

payment = scanner.nextDouble(); scanner.close();

NumberFormat usFormat = NumberFormat.getCurrencyInstance(Locale.US); NumberFormat cnFormat

= NumberFormat.getCurrencyInstance(Locale.CHINA); NumberFormat frFormat =

NumberFormat.getCurrencyInstance(Locale.FRANCE); NumberFormat inFormat =

NumberFormat.getCurrencyInstance(new Locale(&quot;en&quot;, &quot;in&quot;)); String us =
usFormat.format(payment);

String india = inFormat.format(payment); String

france = frFormat.format(payment); String china

= cnFormat.format(payment);

System.out.println(&quot;US: &quot; + us);

System.out.println(&quot;India: &quot; + india);

System.out.println(&quot;China: &quot; + china);

System.out.println(&quot;France: &quot; + france);

}

}
```

Program 4 – Create the dynamic stack in Java by implementing the interfaces that defines push() and pop() methods.

```
package program4;

public interface StackInt {

void push(int item);

int pop();

}

package program4;

public class Stack implements StackInt {

int length;

int stk[]; int

top; Stack()

{

length = 0;

top = -1;

}

Stack(int cap)

{

length = cap;

top = -1;

stk = new int[length];

}

public void push(int item)

{

if(length - 1 == top)

{

int t[] = new int[length * 2];


for(int i = 0; i &lt;= top; i++)

{

t[i] = stk[i];

}

stk = t;

stk[++top] = item;

length = 2* length;
```

```
}
else
stk[++top] = item;
}
public int pop()
{
if(top == -1)
{
System.out.println(&amp;quot;Stack Underflows &amp;quot;);
return -1;
}
else
return stk[top--];
}
public void print()
{
if(top == -1)
{
System.out.println(&amp;quot;Empty Stack&amp;quot;);
return;
}
System.out.print(&amp;quot;[&amp;quot;);
for(int i = 0;i &amp;lt;= top; i++)
{
System.out.print(stk[i] + &amp;quot; &amp;quot;);
}
System.out.println(&amp;quot;]&amp;quot;);
}
}
package program4;
import java.util.*;
public class Main {
public static void main(String[] args)
{
Scanner s = new Scanner(System.in);
int capacity, ch, x;
```

```
Stack st;
System.out.println(&amp;quot;Enter the initial capacity of the stack&amp;quot;);
capacity = s.nextInt();
st = new Stack(capacity);
System.out.print(&amp;quot;Choice:\n 1.Push\n 2.Pop\n 3.Print Stack\n 4.Exit\n&amp;quot;);


while(true)
{
System.out.println(&amp;quot;Enter your choice&amp;quot;);
ch = s.nextInt();
switch (ch)
{
case 1: System.out.println(&amp;quot;Enter the item to be pushed&amp;quot;);
x = s.nextInt();
st.push(x);
break;
case 2: System.out.println(&amp;quot;Popped Element: &amp;quot; + st.pop());
break;
case 3: st.print();
break;
case 4: return;
default: System.out.println(&amp;quot;Invalid choice&amp;quot;);
break;
}
}
}
}
```

Program 5 – Write a Java program to demonstrate the following. A company pays its employees weekly. The employees are of three types.

a) Salaried employees are paid a fixed weekly salary regardless of the number of hours worked.

b) Commission employees are paid a percentage of their sales and

c) Base_salary_plus_commission employees receive a base salary plus percentage of their sales.

For the current pay period, the company has decided a reward for base_salary_plus_commission employees by adding 10 percent to their base salaries. The company wants to implement a Java program that performs its payroll calculations polymorphically. Design and implement the program

using inheritance. Implement the methods to read and write the employee details (like name, emp_id, salary, etc), to compute the employee's gross salary.

```
package program5;

public abstract class Employee {

private String firstName; private

String lastName;

private String emp_id;

// three-argument constructor

public Employee( String first, String last, String ssn )

{

firstName = first;

lastName = last;

emp_id = ssn;

} // end three-argument Employee constructor

// set first name

public void setFirstName( String first )

{

firstName = first; // should validate

} // end method setFirstName

// return first name

public String getFirstName()

{

return firstName;

} // end method getFirstName

// set last name

public void setLastName( String last )


{

lastName = last; // should validate

} // end method setLastName

// return last name

public String getLastName()

{

return lastName;

} // end method getLastName
```

```
// set social security number
public void setEmp_id( String ssn )
{
emp_id = ssn; // should validate
} // end method setSocialSecurityNumber
// return social security number
public String getEmp_id()
{
return emp_id;
} // end method getSocialSecurityNumber
// return String representation of Employee object
@Override
public String toString()
{
return String.format( &quot;%s %s\nsocial security number: %s&quot;,
getFirstName(), getLastName(), getEmp_id() );
} // end method toString
// abstract method overridden by concrete subclasses
public abstract double earnings(); // no implementation here
}
public class Commisioned_Emplyee extends Employee {
private double grossSales; // gross weekly sales
private double commissionRate; // commission percentage
// five-argument constructor
public Commisioned_Emplyee( String first, String last, String ssn, double


sales, double rate )


{
super( first, last, ssn );
setGrossSales( sales );
setCommissionRate( rate );
} // end five-argument CommissionEmployee constructor
// set commission rate
public void setCommissionRate( double rate )
{
```

```
if ( rate &gt; 0.0 &amp;amp;&amp;amp; rate &lt; 1.0 )
commissionRate = rate;
else
throw new IllegalArgumentException( &amp;quot;Commission rate
must be &amp;gt; 0.0 and &amp;lt; 1.0&amp;quot; );
} // end method setCommissionRate

// return commission rate
public double getCommissionRate()
{
return commissionRate;
} // end method getCommissionRate
// set gross sales amount
public void setGrossSales( double sales )
{
if ( sales &amp;gt;= 0.0 )
grossSales = sales;
else
throw new IllegalArgumentException(
&amp;quot;Gross sales must be &amp;gt;= 0.0&amp;quot; );
} // end method setGrossSales
// return gross sales amount public
double getGrossSales()
{
return grossSales;
} // end method getGrossSales
// calculate earnings; override abstract method earnings in Employee
@Override
public double earnings()
{
return getCommissionRate() * getGrossSales();
} // end method earnings
// return String representation of CommissionEmployee object
@Override
public String toString()
{
```

```
return String.format( &quot;%s: %s\n%s: $%,.2f; %s: %.2f&quot;,
&quot;commission employee&quot;, super.toString(),
&quot;gross sales&quot;, getGrossSales(), &quot;commission rate&quot;,
getCommissionRate() );
} // end method toString
}
public class Base_salary_plus_commisssion_employee extends Commisioned_Emplyee {
private double baseSalary; // base salary per week
// six-argument constructor
public Base_salary_plus_commisssion_employee( String first, String last, String ssn, double

sales, double rate, double salary )

{
super( first, last, ssn, sales, rate );
setBaseSalary( salary ); // validate and store base salary
} // end six-argument BasePlusCommissionEmployee constructor
// set base salary
public void setBaseSalary( double salary )
{
if ( salary &gt;= 0.0 )

baseSalary = salary;
else
throw new IllegalArgumentException(
&quot;Base salary must be &gt;= 0.0&quot; );
} // end method setBaseSalary
// return base salary
public double getBaseSalary()
{
return baseSalary;
} // end method getBaseSalary
// calculate earnings; override method earnings in CommissionEmployee
@Override
public double earnings()
{
```

```
return getBaseSalary() + super.earnings();
}// end method earnings
// return String representation of BasePlusCommissionEmployee object
@Override
public String toString()
{
return String.format( &quot;%s %s; %s: $%,.2f&quot;, &quot;base-
salaried&quot;, super.toString(), &quot;base salary&quot;,
getBaseSalary() );
} // end method toString
}
public class Salaried_Employee extends Employee {
private double weeklySalary;
// four-argument constructor
public Salaried_Employee( String first, String last, String ssn, double

salary )

{
super( first, last, ssn ); // pass to Employee constructor
setWeeklySalary( salary ); // validate and store salary
} // end four-argument SalariedEmployee constructor
// set salary
public void setWeeklySalary( double salary )
{
if ( salary &gt;= 0.0 )
weeklySalary = salary;
else
throw new IllegalArgumentException(
&quot;Weekly salary must be &gt;= 0.0&quot; );
} // end method setWeeklySalary
// return salary
public double getWeeklySalary()
{
return weeklySalary;
} // end method getWeeklySalary
```

```
// calculate earnings; override abstract method earnings in Employee
@Override
public double earnings()
{
return getWeeklySalary();
} // end method earnings
// return String representation of SalariedEmployee object
@Override
public String toString()
{
return String.format( &quot;salaried employee: %s\n%s: $%,.2f&quot;,
super.toString(), &quot;weekly salary&quot; , getWeeklySalary() );
} // end method toString
}
import java.util.*;
public class Main {
public static void main(String args[])
{
// create subclass objects
String First_name, Last_name, emp_id;
float salary, rate, sales;
Scanner s = new Scanner(System.in); System.out.println(&quot;Enter First
name of Salaried Employee&quot;); First_name = s.next();
System.out.println(&quot;Enter Last name of Salaried Employee&quot;);
Last_name =s.next();
System.out.println(&quot;Enter Employee id of salaried employee&quot;);
emp_id =s.next();
System.out.println(&quot;Enter salary of salaried employee&quot;);
salary = s.nextFloat(); Salaried_Employee
salariedEmployee =
new Salaried_Employee( First_name, Last_name , emp_id, salary );
System.out.println(&quot;Enter First name of Commissioned Employee&quot;); First_name =
s.next();
System.out.println(&quot;Enter Last name of Commissioned Employee&quot;); Last_name
=s.next();
```

```
System.out.println(&quot;Enter Employee id of Commissioned employee&quot;);
emp_id =s.next();
System.out.println(&quot;Enter sales of Commissioned employee&quot;);
sales = s.nextFloat();
System.out.println(&quot;Enter percentage of sales being included in salary&quot;);
rate = s.nextFloat();
Commisioned_Emplyee commissionEmployee =
new Commisioned_Emplyee(
First_name, Last_name , emp_id, sales, rate); System.out.println(&quot;Enter First
name of base Plus Commission Employee&quot;); First_name = s.next();
System.out.println(&quot;Enter Last name of base Plus Commission Employee&quot;);

Last_name =s.next();
System.out.println(&quot;Enter Employee id of base Plus Commission employee&quot;);
emp_id =s.next();
System.out.println(&quot;Enter sales of base Plus Commission employee&quot;);
sales = s.nextFloat();
System.out.println(&quot;Enter percentage of sales being included in salary&quot;);
rate = s.nextFloat();
System.out.println(&quot;Enter salary of base Plus Commission employee&quot;);
salary = s.nextFloat();
Base_salary_plus_commisssion_employee basePlusCommissionEmployee =
new Base_salary_plus_commisssion_employee( First_name,
Last_name , emp_id, sales, rate, salary );
System.out.println( &quot;Employees processed individually:\n&quot; );
System.out.printf( &quot;%s\n%s: $%,.2f\n\n&quot;,
salariedEmployee, &quot;earned&quot;, salariedEmployee.earnings() );
System.out.printf( &quot;%s\n%s: $%,.2f\n\n&quot;,
commissionEmployee, &quot;earned&quot;, commissionEmployee.earnings() );
System.out.printf( &quot;%s\n%s: $%,.2f\n\n&quot;,
basePlusCommissionEmployee,
&quot;earned&quot;, basePlusCommissionEmployee.earnings() );
// create four-element Employee array
} // end main
}
```

Program 6 –

(a) On a single track two vehicles are running. As vehicles are going in same direction there is no problem. If the vehicles are running in different direction there is a chance of collision. To avoid collisions write a Java program using exception handling. You are free to make necessary assumptions.

(b) Write a Java program to read two integers a and b. Compute a/b and print, when b is not zero. Raise an exception when b is equal to zero.

```
6a)
package program6;
import java.io.*;
class collision extends Exception
{
collision(String s)
{ super(s); }
}
import java.io.DataInputStream;
class _6th
{
public static void main(String args[])
{
String t1=null,t2=null;
try
{
DataInputStream in= new DataInputStream(System.in);
System.out.println(&quot;enter the direction of vehicle1:(left/right):&quot;);
t1=in.readLine();

System.out.println(&quot;enter the direction of vehicle2:(left/right):&quot;);
t2=in.readLine();
if(!t1.equals(t2))
throw new collision(&quot;truck2 has to go on &quot;+ t1 +&quot; direction&quot;);
}
catch(collision e)
{
System.out.println(e);
t2=t1;
```

```
System.out.println(&quot;the collision has been avoided by redirection of truck2&quot;);
}
catch(Exception e)
{ System.out.println(e); }
System.out.println(&quot;direction of truck1 :&quot;+t1);
System.out.println(&quot;direction of truck2 :&quot;+t2);
}
}
```

6b)

```
package program6;
import java.util.Scanner;
class _6th_2nd
{
public static void main(String arg[])
{
while(true) {
try {
// declare and initialize here. int
a, b, c;
Scanner KB = new Scanner(System.in);
// input numbers here.
System.out.print(&quot;Enter first number : &quot;);
a = KB.nextInt();
System.out.print(&quot;Enter second number : &quot;);
b = KB.nextInt();
//throw to catch
c = a / b; System.out.println(&quot;Result:&quot; +
c);
} catch (ArithmeticException e) {
System.out.println(&quot;Error:&quot; + e.getMessage());
System.out.println(&quot;Error:&quot; + e);
}
// here program ends.
System.out.println(&quot;End of Program...&quot;);
}
}
}
```

Program 7 – Write a Java Program to print the multiples of 7, 8 alternatively to demonstrate the concept of inter-thread communication.

```
package program7;
class Seven_Multiple extends Thread
{
int limit;
sharedPrinter printer;
public Seven_Multiple(int limit, sharedPrinter printer)
{
this.limit = limit;
this.printer = printer;
}
@Override public
void run()
{
int mult_of_seven = 7;
while (mult_of_seven &lt;= limit)
{
printer.printSeven(mult_of_seven);
mult_of_seven = mult_of_seven + 7;
}
}
}
class Eight_Multiple extends Thread
{
int limit;
sharedPrinter printer;
public Eight_Multiple(int limit, sharedPrinter printer)
{
this.limit = limit;
this.printer = printer;
}
@Override public
void run()
{
```

```
int mult_of_eight = 8; //First even number is 2 while
(mult_of_eight &lt;= limit)
{
printer.printEight(mult_of_eight); //Calling printEven() method of SharedPrinter class
mult_of_eight = mult_of_eight + 8; //Incrementing to next even number
}
}
}
class sharedPrinter

{
boolean isSevenPrinted = false;
synchronized void printSeven(int number)
{
while (isSevenPrinted)
{
try
{
wait();
}
catch (InterruptedException e)
{
e.printStackTrace();
}
}
System.out.println(Thread.currentThread().getName()+&quot; : &quot;+number);
isSevenPrinted = true;
try
{
Thread.sleep(1000);
}
catch (InterruptedException e)
{
e.printStackTrace();
}
notify();
```

```
}
synchronized void printEight(int number)
{
while (!isSevenPrinted)
{
try
{
wait();
}
catch (InterruptedException e)
{
e.printStackTrace();
}
}
System.out.println(Thread.currentThread().getName()+&amp;quot; : &amp;quot;+number);
isSevenPrinted = false;
try
{
Thread.sleep(1000);
}
catch (InterruptedException e)
{

e.printStackTrace();
}
notify();
}
}
import java.util.Scanner;
public class _7th {
//OddThread to print odd numbers
//Calls printOdd() method of SharedPrinter class until limit is exceeded. public static
void main(String[] args)
{
Scanner s1 = new Scanner(System.in);
System.out.print(&amp;quot;Enter the extent to which the thread have to be printed : &amp;quot;);
```

```
int n = s1.nextInt();

sharedPrinter printer = new sharedPrinter();

Seven_Multiple sevenMultiple = new Seven_Multiple(n, printer);

sevenMultiple.setName(&amp;quot;sevenThread&amp;quot;);

Eight_Multiple eightMultiple = new Eight_Multiple(n, printer);

eightMultiple.setName(&amp;quot;eightThread&amp;quot;);

sevenMultiple.start(); eightMultiple.start();

}

}
```

Program 8 – Write a program that reads in a series of first names and eliminates duplicates by storing them in a Set. Allow the user to search for a first name.

```
package program8;

import java.util.*;

public class _8th {

public static void main(String[] args) {

// Write a program thats ask for first names and store it in an array.

String fName;

Scanner input = new Scanner(System.in);

System.out.print(&amp;quot;Enter the number of names :&amp;quot;); int

number = input.nextInt();

System.out.println(&amp;quot;Enter the &amp;quot; + number + &amp;quot; of names one by one &amp;quot;);

List&amp;lt;String&amp;gt; list = new ArrayList&amp;lt;&amp;gt;();

for (int i = 0; i &amp;lt; number; i++) {

System.out.print(&amp;quot;Enter First Name: &amp;quot;);

list.add(input.next());

}

System.out.println(&amp;quot;Initial Array Elements :&amp;quot;);

System.out.printf(&amp;quot;%s &amp;quot;, list); System.out.println();

System.out.println(&amp;quot;After removing duplicates method :&amp;quot;);

removeDuplicates(list);

System.out.println();


searchForName(list);
```

```
}
private static void removeDuplicates(Collection&lt;String&gt; values) {
Set&lt;String&gt; set = new HashSet&lt;String&gt;(values); System.out.printf(&quot;%s &quot;,
set);
System.out.println();
}
public static void searchForName(Collection&lt;String&gt; names) { String
someName = &quot;&quot;;
Set&lt;String&gt; set = new HashSet&lt;String&gt;(names); Scanner
input = new Scanner(System.in);
System.out.print(&quot;Search a name: &quot;);
someName = input.nextLine();
if (set.contains(someName)) {
System.out.println(&quot;set contains name&quot;);
} else {
System.out.println(&quot;set doesn&#39;t contain this name&quot;);
}
}
}
```

Program 9 –

(a) Write the program to count occurrences of a given character using Regex in Java.

(b) Write a program to check if a string contains only alphabets in Java using Regex.

```
9a) package program9;
import java.util.Scanner; import
java.util.regex.Matcher; import
java.util.regex.Pattern; public class
_9th_1st {
public static long count(String s, char ch)
{
// Use Matcher class of java.util.regex
// to match the character
Matcher matcher
```

```
= Pattern.compile(String.valueOf(ch))
.matcher(s);
int res = 0;
// for every presence of character ch
// increment the counter res by 1 while
(matcher.find()) {
res++;
}
return res;
}
// Driver method
public static void main(String args[])
{
Scanner s1 = new Scanner(System.in);

System.out.print(&quot;Enter the text :&quot;); String
str = s1.next();
System.out.print(&quot;Enter the character to be found :&quot;);
char c = s1.next().charAt(0);
System.out.print(&quot;The following is the occurances of the character :&quot;);
System.out.println(count(str, c));
}
}
```

9b)

```
package program9;
import java.util.Scanner;
public class _9th_2nd {
public static boolean isStringOnlyAlphabet(String str)
{
return ((str != null)
&amp;&amp; (!str.equals(&quot;&quot;))
&amp;&amp; (str.matches(&quot;^[a-zA-Z]*$&quot;)));
}
// Main method
public static void main(String[] args)
{
```

```
// Checking for True case
Scanner s1 = new Scanner(System.in);
System.out.print(&amp;quot;Enter the string :&amp;quot;); String
getit = s1.next();
System.out.print(&amp;quot;The boolean value below tells whether string contains only alphabets or
not:\n&amp;quot;);
System.out.println(&amp;quot;Output: &amp;quot; + isStringOnlyAlphabet(getit));
}
}
```

Program 10 – Write a program to Check if characters of a given string can be rearranged to form a palindrome.

```
package program10;
import java.util.Scanner;
public class _10th {
static int NO_OF_CHARS = 256;
/* function to check whether characters of a
string can form a palindrome */
static boolean canFormPalindrome(String str) {
// Create a count array and initialize all
// values as 0
int count[] = new int[NO_OF_CHARS];
for( int i = 0 ; i&amp;lt; count.length;i++)
{
count[i] = 0;

}
// For each character in input strings,
// increment count in the corresponding
// count array
for (int i = 0; i &amp;lt; str.length(); i++)
count[(int)(str.charAt(i))]++;
// Count odd occurring characters int
odd = 0;
for (int i = 0; i &amp;lt; NO_OF_CHARS; i++)
```

```
{
if ((count[i] &amp;amp; 1) == 1)
odd++;
if (odd &gt; 1)
return false;
}
// Return true if odd count is 0 or 1,
return true;
}
// Driver program
public static void main(String args[])
{
:&amp;quot;);


}
System.out.print(&amp;quot;Enter the string and check whether it can form palindrome on rearranging
if (canFormPalindrome(new Scanner(System.in).next()))
System.out.println(&amp;quot;Yes&amp;quot;);
else
System.out.println(&amp;quot;No&amp;quot;);
}
```