**1.WAP to pass a null to specified method**

**public** **class** PassingNullOnMethod {

**public** **static** **void** doSomething(Object obj) {

System.*out*.println("Object called");

}

**public** **static** **void** doSomething(**char**[] obj) {

System.*out*.println("Array called");

}

**public** **static** **void** doSomething(Integer obj) {

System.*out*.println("Integer called");

}

**public** **static** **void** main(String[] args) {

// doSomething(null); //Exception in thread "main" java.lang.Error: Unresolved

compilation problem:

*doSomething*("null"); // Object called

*doSomething*('c'); // Array called

*doSomething*(45); // Integer called

}

}

**WAP to check String when passing null and others**

**public** **class** Test {

**public** **static** **void** main(String[] args) {

String s1 = **null**;

String s2 = "abc";

String s3 = **new** String("abc");

System.*out*.println(s1 == s2); // false

System.*out*.println(s1 == s3); // false

System.*out*.println(s2 == s3); // false

System.*out*.println(s2.equals(s3)); // false

System.*out*.println(s1.equals(s2)); // NullPointerException

System.*out*.println(s1.equals(s3)); // NullPointerException

}

}

**WAP to split the word according to specified word**

**public** **class** SplittingTheWordAndCount {

**public** **static** **void** main(String[] args) {

**int** count = 0;

String string = "This is india and I am INDIAN";

String[] word = string.split(" ");

**for** (String s1 : word) {

System.*out*.println(s1);

count++;

}

System.*out*.println(count);

}

}

**WAP to reverse a given String.**

**public** **class** StringReverse {

**public** **static** **void** main(String[] args) {

String s = "Hello World";

String revString = "";

**for** (**int** i = s.length()-1; i >= 0; --i) {

System.*out*.println(i);

revString = revString + s.charAt(i);

}

System.*out*.println(revString);

}

}

Output:dlroW olleH

**WAP to reverse a string and print word by word.**

**public** **class** ReverseStringByWord {

**public** **static** **void** main(String[] args) {

String s = "He is the one";

String finalStr = "";

String s2[] = s.split(" ");

**for** (**int** i = s2.length - 1; i >= 0; --i) {

finalStr += s2[i] + " ";

}

System.*out*.println(finalStr);

}

} **Output: one the is He**

**WAP to reverse a string and check it palindrome or not.**

**public** **class** PalindromeDemo {

**public** **static** **void** main(String[] args) {

String str = "MADAM";

String revstring = "";

**for** (**int** i = str.length() - 1; i >= 0; --i) {

revstring += str.charAt(i);

}

System.*out*.println(revstring);

**if** (revstring.equalsIgnoreCase(str)) {

System.*out*.println("The string is Palindrome");

} **else** {

System.*out*.println("Not Palindrome");

}

}

} output:- MADAM, Palindrome

**2.WAP to reverse a number:**

**public** **class** Reverse {

**public** **static** **void** main(String[] args) {

**int** number = 12345;

**int** reverseNumber = 0;

**int** temp = 0;

**while**(number > 0) {

temp = number % 10;

reverseNumber = reverseNumber \* 10 + temp;

number = number / 10;

}

System.*out*.println("Your reverse numbe:"+reverseNumber);

}

} OUTPUT:54321

**3.WAP to find a duplicate string:**

**public** **class** DuplicatesInArray {

**public** **static** **void** main(String[] args) {

String[] s = {"abc", "def", "mno", "xyz", "pqr", "xyz", "def"};

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

**for** (**int** j = i+1; j < s.length; j++) {

**if**( (s[i].equals(s[j])) ) {

System.*out*.println("Duplicate Element is : "+s[j]);

}

}

}

}

}

Output:Duplicate Element is : def

Duplicate Element is : xyz

**4.WAP to find duplicate from two string:**

**public** **class** CommonElementsBetweenTwoArrays {

**public** **static** **void** main(String[] args) {

String[] s1 = {"ONE", "TWO", "THREE", "FOUR", "FIVE", "FOUR"};

String[] s2 = {"THREE", "FOUR", "FIVE", "SIX", "SEVEN", "FOUR"};

HashSet<String> set = **new** HashSet<String>();

**for** (**int** i = 0; i < s1.length; i++) {

**for** (**int** j = 0; j < s2.length; j++) {

**if**(s1[i].equals(s2[j])) {

set.add(s1[i]);

}

}

}

System.*out*.println(set);

}

}

OUTPUT : [THREE, FOUR, FIVE]

**5.WAP to find missing number from an array:**

**public** **class** FindMissingNumber {

**private** **static** **int** calculateSum(**int**[] numbers) {

**int** sum = 0;

**for** (**int** n : numbers)

sum += n;

System.*out*.println(":"+sum);

}

**return** sum;

}

**public** **static** **void** main(String[] args) {

// 4 is missing

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

**int** N = 5;

**int** idealSum = (N \* (N + 1)) / 2;

**int** sum = *calculateSum*(numbers);

**int** missingNumber = idealSum - sum;

System.*out*.println(“MissingNumber:” + missingNumber);

}

}

OUTPUT: MissingNumber:4

**6.WAP to find MAXIMUM and MINIMUM number from the array.**

**public** **class** FindMaxNumFromArray {

**public** **static** **void** main(String[] args) {

// array of 10 numbers

**int** numbers[] = **new** **int**[] { 32, 43, 53, 54, 32 };

// assign first element of an array to largest and smallest

**int** smallest = numbers[0];

**int** largetst = numbers[0];

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

**if** (numbers[i] > largetst) {

largetst = numbers[i];

}

**else** **if** (numbers[i] < smallest) {

smallest = numbers[i];

}

}

System.*out*.println("Largest Number is : " + largetst);

System.*out*.println("Smallest Number is : " + smallest);

}

}

OUTPUT: Largest Number is : 54

Smallest Number is : 32

**6.WAP to find middle element from the ArrayList:**

**public** **class** FindMiddleElement {

**public** **static** **void** main(String[] args) {

ArrayList list = **new** ArrayList();

**for** (**int** i = 0; i < 10; i++) {

list.add(i);

}

System.*out*.println(list);

**int** middle = list.size() / 2;

System.*out*.println(“Middle Element:” + middle);

}

}

OUTPUT: Middle Element: 5

**8.WAP to find number from given alphanumeric string and add them.**

**public** **class** FindNumberFromStringToAdd {

**public** **static** **void** main(String[] args) {

String a="jklmn135pjro25ops";

**int** sum = 0;

String num = "";

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

**if**(Character.*isDigit*(a.charAt(i))) {

num = num + a.charAt(i);

} **else** {

**if**(!num.equals("")) {

sum = sum + Integer.*parseInt*(num);

num = "";

}

}

}

System.*out*.println(“Found number and addtion:” + sum);

}

}

OUTPUT:Found number and addtion:160

**9.WAP to find first and second largest number from the given array.**

**class** FindSecondLargestNumber {

**public** **static** **void** main(String args[]) {

**int** numbers[] = { 6, 3, 37, 12, 46, 5, 64, 21 };

**int** highest = 0;

**int** second\_highest = 0;

**for** (**int** n : numbers) {

**if** (highest < n) {

second\_highest = highest;

highest = n;

}

**else** **if** (second\_highest < n) {

second\_highest = n;

}

}

System.*out*.println("First Max Number: " + highest);

System.*out*.println("Second Max Number: " + second\_highest);

}

}

OUTPUT: First Max Number: 64

Second Max Number: 46

2.WAP to generate prime number up to 100.

**public** **class** PrimeNumber {

**public** **static** **void** main(String[] args) {

**for** (**int** i = 2; i <= 100; i++) {

**for** (**int** j = 2; j <= i; j++) {

**if**(j==i) {

System.*out*.println(i);

}

**if**(i%j==0) {

**break**;

}

}

}

}

}

OUTPUT: 2 3 5 7 11 13 17 so on

3. WAP to print Fabonacci Series

**public** **class** FabonacciSeries {

**public** **static** **void** main(String[] args) {

**int** a = 0, b = 1, c;

System.*out*.println(a);

System.*out*.println(b);

c = a + b;

**while** (c < 89) {

a = b;

b = c;

c = a + b;

System.*out*.print(c);

}

}

}

OUTPUT: 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

**2.WAP to create custom immutable class.**

**public** **final** **class** Employee {

**final** String pancardNumber;

**public** Employee(String pancardNumber) {

**this**.pancardNumber = pancardNumber;

}

**public** String getPancardNumber() {

**return** pancardNumber;

}

}

Client class to make call

**public** **class** EmployeeUser {

**public** **static** **void** main(String[] args) {

Employee e1 = **new** Employee("12345");

System.*out*.println(e1.getPancardNumber());//1234

Employee e2 = **new** Employee("9999"); //9999

System.*out*.println(e2.getPancardNumber());

System.*out*.println(e1.equals(e2)); //false

System.*out*.println(e1==e2); //false

System.*out*.println(e1.hashCode()); //31168322

System.*out*.println(e2.hashCode()); //17225372

}

}

**WAP to generate Random Number: 0…99**

**public** **final** **class** RandomInteger {

**public** **static** **final** **void** main(String... aArgs) {

Random randomGenerator = **new** Random();

**for** (**int** idx = 1; idx <= 10; ++idx) {

**int** randomInt = randomGenerator.nextInt(100);

System.*out*.println("Generated : " + randomInt);

}

}

}

5. WAP to find Factorial of given number:

6. WAP to find temprutere of the given number.

7. WAP to Play with Arrays.

8. WAP to demonstrate Polymorphism( Compile time Polumorphism, Runtime Time Polymorphism)

**9.WAP to avoid duplicate objects as a keys into map**

**public** **class** DefinedObjectAsAKeyInHashMap {

**public** **static** **void** main(String a[]) {

HashMap<Price, String> hm = **new** HashMap<Price, String>();

hm.put(**new** Price("Banana", 20), "Banana");

hm.put(**new** Price("Apple", 40), "Apple");

hm.put(**new** Price("Orange", 30), "Orange");

*printMap*(hm);

Price key = **new** Price("Banana", 20);

System.*out*.println("Does key available? " + hm.containsKey(key));

}

**public** **static** **void** printMap(HashMap<Price, String> map) {

Set<Price> keys = map.keySet();

**for** (Price p : keys) {

System.*out*.println(p + "==>" + map.get(p));

}

}

}

**class** Price {

**private** String item;

**private** **int** price;

**public** Price(String itm, **int** pr) {

**this**.item = itm;

**this**.price = pr;

}

**//Setter and Getter**

**public** String toString() {

**return** "item: " + item + " price: " + price;

}

**public** **int** hashCode() {

System.*out*.println("In hashcode");

**int** hashcode = 0;

hashcode = price \* 20;

hashcode += item.hashCode();

**return** hashcode;

}

**public** **boolean** equals(Object obj) {

System.*out*.println("In equals");

**if** (obj **instanceof** Price) {

Price pp = (Price) obj;

**return** (pp.item.equals(**this**.item) && pp.price == **this**.price);

} **else** {

**return** **false**;

}

}

}

**10.WAP to sort the object into collection.**

**public** **class** TreeMapDemo {

**public** **static** **void** main(String[] args) {

TreeMap t = **new** TreeMap( **new** MyComprator());

t.put(102, "ram");

t.put(101, "nit");

t.put(104, "kar"); //Homogeneous key

t.put("104x", "kar"); //Heterogeneous comparison

System.*out*.println(t);

}

}

**class** MyComprator **implements** Comparator {

@Override

**public** **int** compare(Object obj1, Object obj2) {

String s1 = obj1.toString();

String s2 = obj2.toString();

**return** s1.compareTo(s2);

//return -s1.compareTo(s2);

}

}

OUTPUT: {101=nit, 102=ram, 104=kar, 104x=kar}

**11.WAP to sort based on id, name, weight**

**class** Person {

String name;

**int** age;

**double** weight;

Person(String name, **int** age, **double** weight) {

**this**.name = name;

**this**.age = age;

**this**.weight = weight;

}

**public** String toString() {

**return** "(" + name + "," + age + "," + weight + ")";

}

**static** **class** SortBasedOnName **implements** Comparator {

@Override

**public** **int** compare(Object o1, Object o2) {

Person p1 = (Person) o1;

Person p2 = (Person) o2;

**return** p1.name.compareTo(p2.name);

}

}

**static** **class** SortBasedOnAge **implements** Comparator {

@Override

**public** **int** compare(Object o1, Object o2) {

Person p1 = (Person) o1;

Person p2 = (Person) o2;

**return** p1.age - p2.age;

}

}

**static** **class** SortBasedOnWeight **implements** Comparator {

@Override

**public** **int** compare(Object o1, Object o2) {

Person p1 = (Person) o1;

Person p2 = (Person) o2;

**return** (**int**) ((**int**) p1.weight - p2.weight);

}

}

}

**public** **class** Manager8 {

**public** **static** **void** main(String[] args) {

Person[] x = {

**new** Person("Ramu", 22, 55.09),

**new** Person("Poonam", 4, 35.09),

**new** Person("Aukesh", 67, 05.88),

**new** Person("Khushboo", 23, 50.09),

**new** Person("Babita", 89, 25.09)

};

System.*out*.println("your Data : " + Arrays.*toString*(x));

Arrays.*sort*(x, **new** Person.SortBasedOnName());

System.*out*.println("Based on Name : " + Arrays.*toString*(x));

Arrays.*sort*(x, **new** Person.SortBasedOnAge());

System.*out*.println("Based on Age : " + Arrays.*toString*(x));

Arrays.*sort*(x, **new** Person.SortBasedOnWeight());

System.*out*.println("Based on Weight : " + Arrays.*toString*(x));

}

}

Output:

your Data : [(Ramu,22,55.09), (Poonam,4,35.09), (Aukesh,67,5.88), (Khushboo,23,50.09), (Babita,89,25.09)]

Based on Name : [(Aukesh,67,5.88), (Babita,89,25.09), (Khushboo,23,50.09), (Poonam,4,35.09), (Ramu,22,55.09)]

Based on Age : [(Poonam,4,35.09), (Ramu,22,55.09), (Khushboo,23,50.09), (Aukesh,67,5.88), (Babita,89,25.09)]

Based on Weight : [(Aukesh,67,5.88), (Babita,89,25.09), (Poonam,4,35.09), (Khushboo,23,50.09), (Ramu,22,55.09)]

**12.WAP to create own custom Immutable class:**

**final** **public** **class** Test {

**final** **private** **int** i;

**public** Test(**int** i) {

**this**.i = i;

}

**public** Test modify(**int** i) {

**if**(**this**.i == i ) {

**return** **this**;

} **else** {

**return** **new** Test(i);

}

}

}

**public** **class** TestClient {

**public** **static** **void** main(String[] args) {

Test t1 = **new** Test(12);

Test t2 = t1.modify(15);

Test t3 = t1.modify(12);

System.*out*.println(t1==t2); //false

System.*out*.println(t1==t3); //true

}

}

OUTPUT: false

True

**Another example**

**public** **final** **class** Employee {

**final** String pancardNumber;

**public** Employee(String pancardNumber) {

**this**.pancardNumber = pancardNumber;

}

**public** String getPancardNumber() {

**return** pancardNumber;

}

}

**public** **class** EmployeeUser {

**public** **static** **void** main(String[] args) {

Employee e1 = **new** Employee("12345");

System.*out*.println(e1.getPancardNumber());

Employee e2 = **new** Employee("9999");

//e2.pancardNumber = "11111"; we cant perform any change

System.*out*.println(e2.getPancardNumber());

System.*out*.println(e1.equals(e2));

System.*out*.println(e1==e2);

System.*out*.println(e1.hashCode());

System.*out*.println(e2.hashCode());

}

}

OUTPUT:12345

9999

false

false

31168322

17225372

**13. WAP to create singleton object without thread safety.**

**public** **class** SingletonWithoutThreadSafety {

**private** **static** SingletonWithoutThreadSafety *instance* = **null**;

**private** SingletonWithoutThreadSafety() {

}

**public** **static** SingletonWithoutThreadSafety getInstance() {

**if** (*instance* == **null**) {

*instance* = **new** SingletonWithoutThreadSafety();

}

**return** *instance*;

}

}

**With thread safety**

**public** **class** SingletonWithThreadSafety

{

**private** **static** SingletonWithThreadSafety *instance*= **null**;

**private** **static** Object *mutex*= **new** Object();

**private** SingletonWithThreadSafety()

{

}

**public** **static** SingletonWithThreadSafety getInstance()

{

**if**(*instance*==**null**)

{

**synchronized** (*mutex*)//achiving thread safety

{

**if**(*instance*==**null**) *instance*= **new** SingletonWithThreadSafety();

}

}

**return** *instance*;

}

}

**WAP to create five objects for any class**

**class** Sample

{

**private** **static** **int** *i* = 0;

**private** Sample()

{

}

**public** **static** Sample CreateInstance() **throws** Exception

{

**if** (*i* < 5)

{

*i*++;

**return** **new** Sample();

}

**else**

{

**throw** **new** Exception("Can not create more then 5 instance of this class");

}

}

}

**public** **class** FivetonObject

{

**public** **static** **void** main(String[] args) **throws** Exception

{

Sample s1 = Sample.*CreateInstance*();

Sample s2 = Sample.*CreateInstance*();

Sample s3 = Sample.*CreateInstance*();

Sample s4 = Sample.*CreateInstance*();

Sample s5 = Sample.*CreateInstance*();

Sample s6 = Sample.*CreateInstance*();// When we try creating 6th objects will get exception

}

}

**14. WAP to remove string from the given string:**

**public** **class** RemoveStringFromGiven

{

**public** **static** **void** main(String[] args)

{

String s1 = "this is java";

String s2 = "java";

StringTokenizer st = **new** StringTokenizer(s1);

System.*out*.println(st.toString());

**if**(st.equals(s2))

{

s1.replace(s2, "");

}

**else**

{

System.*out*.println(st);

}

}

}

**StringTokenizer using multiple**

**public** **class** App

{

**public** **static** **void** main(String[] args)

{

String str = "This is String , split by StringTokenizer, created by mkyong";

StringTokenizer st = **new** StringTokenizer(str);

System.*out*.println("---- Split by space ------");

**while** (st.hasMoreElements()) {

System.*out*.println(st.nextElement());

}

System.*out*.println("---- Split by comma ',' ------");

StringTokenizer st2 = **new** StringTokenizer(str, ",");

**while** (st2.hasMoreElements()) {

System.*out*.println(st2.nextElement());

}

}

}

---- Split by space ------

This

is

String

,

split

by

StringTokenizer,

created

by

mkyong

---- Split by comma ',' ------

This is String

split by StringTokenizer

created by mkyong

**14.WAP to serialize and deserialize the object:**

**public** **class** Student **implements** Serializable

{

**int** id;

String name;

//transient String address //will not be serialized

**public** Student(**int** id, String name)

{

**this**.id = id;

**this**.name = name;

}

}

Serialize

**class** Persist

{

**public** **static** **void** main(String args[])**throws** Exception

{

Student s1 =**new** Student(211,"Priyadarshi");

FileOutputStream fout=**new** FileOutputStream("f.txt");

//FileOutputStream fout=new FileOutputStream("D:\\f.txt");//location to drive

ObjectOutputStream out=**new** ObjectOutputStream(fout);

out.writeObject(s1);

out.flush();

System.*out*.println("success");

}

}

Deserialize

**class** Depersist

{

**public** **static** **void** main(String args[])**throws** Exception

{

ObjectInputStream in = **new** ObjectInputStream(**new** FileInputStream("f.txt"));

Student s = (Student)in.readObject();

System.*out*.println(s.id + " " + s.name);

in.close();

}

}

**WAP to print Odd and Even number from two thread.**

**class** PrintOdd **implements** Runnable {

**public** **static** **boolean** *oddFlag* = **true**;

**public** **void** run() {

**for** (**int** i = 1; i <= 9;) {

**if** (*oddFlag*) {

System.*out*.print(i + " ");

*oddFlag* = **false**;

i = i + 2;

}

}

}

}

**class** PrintEven **implements** Runnable {

**public** **void** run() {

**for** (**int** i = 2; i <= 10;) {

**if** (!PrintOdd.*oddFlag*) {

System.*out*.print(i + " ");

PrintOdd.*oddFlag* = **true**;

i = i + 2;

}

}

}

}

**public** **class** PrintNaturalFromTwoThread {

**public** **static** **void** main(String[] args) {

PrintEven prEven = **new** PrintEven();

PrintOdd prOdd = **new** PrintOdd();

Thread evenThread = **new** Thread(prEven);

Thread oddThread = **new** Thread(prOdd);

evenThread.start();

oddThread.start();

}

}

Output: **1 2 3 4 5 6 7 8 9 10**