**Array: Basic example**

**package** arrays;

**public** **class** ArrayExample {

}

// Pass array as object in method.

/\* public static int getSum(int nums[]) {

int sum = 0;

for(int val : nums)

//sum = sum + val;

sum += val;

return sum;

}

public static void main (String[] args) {

int values[] = {45,76,34,78,223};

int total = getSum(values);

System.out.println(total);

}

}\*/

// Basic of array

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

int values[] = {45,76,34,78,223};

int size = values.length;

for (int index =0; index<size; index++) {

System.out.println(values[index]);

}

for(int val : values)

//System.out.println(values[val]);

System.out.println(values);

}

}

\*/

**Another Example of Array:**

**package** arrays;

**import** dryfruit.Fruit;

**public** **class** ArrayExampleObject {

**private** **static** **void** showFruitNames(Fruit fruitData[]) {

**for** (Fruit fr :fruitData ) {

System.***out***.println(fr.getName().toUpperCase());

}

}

**private** **static** String[] getFruitNames(Fruit fruitData[]) {

**int** size = fruitData.length;

String fruitNames[] = **new** String[size];

**int** index = 0;

**for**( Fruit fr : fruitData) {

String name = fr.getName().toUpperCase();

fruitNames[index] = name;

index++;

}

**return** fruitNames;

}

**public** **static** String[] getfruitcolors(Fruit getfrcolr[]) {

**int** size = getfrcolr.length;

String getFruitColors[] = **new** String[size];

**int** index = 0;

**for**(Fruit fr : getfrcolr ) {

String color = fr.getColor().toUpperCase();

getFruitColors[index] = color;

index++;

}

**return** getFruitColors;

}

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

Fruit fruits[] = **new** Fruit[3];

fruits[0] = **new** Fruit();

fruits[1] = **new** Fruit("Apple", "Red");

fruits[2] = **new** Fruit("Orange", "Orange");

String fName[] = *getFruitNames*(fruits);

**for** (String fname : fName)

System.***out***.println(fname);

String fcolor[] = *getfruitcolors*(fruits);

**for** (String fColor : fcolor)

System.***out***.println(fColor);

System.***out***.println(fruits[0].getName());

System.***out***.println(fruits[0].getColor());

System.***out***.println(fruits[1].getName());

System.***out***.println(fruits[1].getColor());

System.***out***.println(fruits[2].getName());

System.***out***.println(fruits[2].getColor());

*showFruitNames*(fruits);

/\* fruits f1 = new Fruit();

fruits f2 = new Fruit("Apple", "Red");

fruits f3 = new Fruit("Orange", "Orange");

Fruit FruitBasket[] = {f1, f2,f3};\*/

}

}

**Fruit Class :**

**package** dryfruit;

**public** **class** Fruit {

**private** String name, color;

**public** Fruit() {

name = "mango";

color = "yellow";

}

**public** Fruit(String name, String color) {

**super**();

**this**.name = name;

**this**.color = color;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

}

**FruitMain :**

**package** dryfruit;

**public** **class** FruitMain {

**public** **static** Fruit changeFruit(Fruit oldFruit) {

oldFruit.setName("Apple");

oldFruit.setColor("Red");

**return** oldFruit;

}

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

Fruit f1 = **new** Fruit();

//Printing value before change.

System.***out***.println(f1.getName());

System.***out***.println(f1.getColor());

Fruit changedFruit = *changeFruit*(f1);

//Printing value after change.

System.***out***.println(changedFruit.getName());

System.***out***.println(changedFruit.getColor());

}

}

/\*

public static void changeFruit(Fruit oldFruit) {

oldFruit.setName("Apple");

oldFruit.setColor("Red");

}

public static void main(String[] args) {

Fruit f1 = new Fruit();

//Printing value before change.

System.out.println(f1.getName());

System.out.println(f1.getColor());

changeFruit(f1);

//Printing value after change.

System.out.println(f1.getName());

System.out.println(f1.getColor());

}

}\*/

**Containment:**

**package** containment;

**public** **class** Contact {

**private** String mobileNo, emailAddress;

**public** Contact() {

mobileNo = "45656564";

emailAddress = "sdsdf@gmail.com";

}

**public** Contact(String mobileNo, String emailAddress) {

**this**.mobileNo = mobileNo;

**this**.emailAddress = emailAddress;

}

**public** String getMobileNo() {

**return** mobileNo;

}

**public** **void** setMobileNo(String mobileNo) {

**this**.mobileNo = mobileNo;

}

**public** String getEmailAddress() {

**return** emailAddress;

}

**public** **void** setEmailAddress(String emailAddress) {

**this**.emailAddress = emailAddress;

}

}

**package** containment;

**public** **class** CreditCard {

**private** String cardNo, bank;

**private** **int** creditLimit;

**public** CreditCard() {

cardNo = "46466";

bank = "HDFC";

creditLimit = 75000;

}

**public** CreditCard(String cardNo, String bank, **int** creditLimit) {

**this**.cardNo = cardNo;

**this**.bank = bank;

**this**.creditLimit = creditLimit;

}

**public** String getCardNo() {

**return** cardNo;

}

**public** **void** setCardNo(String cardNo) {

**this**.cardNo = cardNo;

}

**public** String getBank() {

**return** bank;

}

**public** **void** setBank(String bank) {

**this**.bank = bank;

}

**public** **int** getCreditLimit() {

**return** creditLimit;

}

**public** **void** setCreditLimit(**int** creditLimit) {

**this**.creditLimit = creditLimit;

}

}

**package** containment;

**public** **class** Customer {

**private** **int** customerId;

**private** String name;

**private** Contact contactDetails; // object of another class

**private** CreditCard cardDetails; // object of another class

**public** Customer() {

customerId = 1001;

name = "Dipti";

contactDetails = **new** Contact();

}

**public** Customer(**int** customerId, String name, Contact contactDetails, CreditCard cardDetails) {

**this**.customerId = customerId;

**this**.name = name;

**this**.contactDetails = contactDetails;

**this**.cardDetails = cardDetails;

}

**public** **int** getCustomerId() {

**return** customerId;

}

**public** **void** setCustomerId(**int** customerId) {

**this**.customerId = customerId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Contact getContactDetails() {

**return** contactDetails;

}

**public** **void** setContactDetails(Contact contactDetails) {

**this**.contactDetails = contactDetails;

}

**public** CreditCard getCardDetails() {

**return** cardDetails;

}

**public** **void** setCardDetails(CreditCard cardDetails) {

**this**.cardDetails = cardDetails;

}

}

**package** containment;

**public** **class** CustomerMain {

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

// creating customer without credit card.

Customer c1 = **new** Customer();

// Creating a customer with credit card.

Contact contact2 = **new** Contact("4535345", "ere@gmail.com");

CreditCard creditcard2 = **new** CreditCard("5453", "HSBC", 60000);

Customer c2 = **new** Customer(1002, "Bill", contact2, creditcard2);

Contact ct = c1.getContactDetails();

System.***out***.println(ct.getEmailAddress());

String ct2 = c1.getContactDetails().getEmailAddress();

System.***out***.println(ct2);

System.***out***.println(c2.getContactDetails().getEmailAddress());

CreditCard cc = c1.getCardDetails();

**if** (cc != **null**)

System.***out***.println(cc.getCreditLimit());

**else**

System.***out***.println("No Credit card");

}

}

1. **Single Inheritance :**

**package** Inheritence;

**public** **class** Player {

**private** String name;

**private** **int** age;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

}

**package** Inheritence;

**public** **class** CricketPayer **extends** Player {

**private** **int** runs;

**public** **int** getRuns() {

**return** runs;

}

**public** **void** setRuns(**int** runs) {

**this**.runs = runs;

}

}

**package** Inheritence;

**public** **class** PlayerMain {

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

CricketPayer cp = **new** CricketPayer();

cp.setName("Sachin T");

cp.setAge(55);

cp.setRuns(145366);

}

}

**2. Single Inheritance with constructor: Invoke – down to top , execution top to bottom**

**package** Inheritence;

**public** **class** Player {

**private** String name;

**private** **int** age;

**public** Player() {

name = "Sania";

age = 50;

System.***out***.println("In Player Class");

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

}

**package** Inheritence;

**public** **class** CricketPayer **extends** Player {

**private** **int** runs;

**public** CricketPayer() {

runs = 5000;

System.***out***.println("In CricketPlayer Class");

}

**public** **int** getRuns() {

**return** runs;

}

**public** **void** setRuns(**int** runs) {

**this**.runs = runs;

}

}

**package** Inheritence;

**public** **class** PlayerMain {

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

CricketPayer cp = **new** CricketPayer();

/\* cp.setName("Sachin T");

cp.setAge(55);

cp.setRuns(145366);\*/

System.***out***.println(cp.getName());

System.***out***.println(cp.getAge());

System.***out***.println(cp.getRuns());

}

}

**Super Keyword with parameterized constructor:**

**package** Inheritence;

**public** **class** Player {

**private** String name;

**private** **int** age;

**public** Player() {

name = "Sania";

age = 50;

System.***out***.println("In Player Class");

}

**public** Player(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

**public** Player(**int** age, String name) {

**this**.age = age;

**this**.name = name;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

}

**package** Inheritence;

**public** **class** CricketPayer **extends** Player {

**private** **int** runs;

**public** CricketPayer() {

runs = 5000;

System.***out***.println("In CricketPlayer Class");

}

**public** CricketPayer(String name, **int** age, **int** runs) {

**super**(name, age);

**this**.runs = runs;

}

**public** **int** getRuns() {

**return** runs;

}

**public** **void** setRuns(**int** runs) {

**this**.runs = runs;

}

}

**package** Inheritence;

**public** **class** PlayerMain {

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

CricketPayer cp = **new** CricketPayer();

CricketPayer cp1 = **new** CricketPayer("Dhoni", 50, 34300);

/\* cp.setName("Sachin T");

cp.setAge(55);

cp.setRuns(145366);\*/

System.***out***.println(cp.getName());

System.***out***.println(cp.getAge());

System.***out***.println(cp.getRuns());

System.***out***.println(cp1.getName());

System.***out***.println(cp1.getAge());

System.***out***.println(cp1.getRuns());

}

}

Polymorphism : Method Overriding.

**package** overriding;

**public** **class** Shape {

**private** String name;

**public** Shape() {

name = "shape1";

}

**public** Shape(String name) {

**this**.name = name;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **float** getArea() {

**return** 0.0f;

}

}

**package** overriding;

**public** **class** Rectangle **extends** Shape {

**private** **float** lenght, breadth;

**public** Rectangle() {

lenght = 5;

breadth = 6;

}

**public** Rectangle(String name, **float** lenght, **float** breadth) {

**super**(name);

**this**.lenght = lenght;

**this**.breadth = breadth;

}

**public** **float** getLenght() {

**return** lenght;

}

**public** **void** setLenght(**float** lenght) {

**this**.lenght = lenght;

}

**public** **float** getBreadth() {

**return** breadth;

}

**public** **void** setBreadth(**float** breadth) {

**this**.breadth = breadth;

}

**public** **float** getArea() {

**return** lenght \* breadth;

}

}

**package** overriding;

**public** **class** Circle **extends** Shape {

**private** **float** radius;

**public** Circle() {

radius = 4;

}

**public** **float** getRadius() {

**return** radius;

}

**public** **void** setRadius(**float** radius) {

**this**.radius = radius;

}

**public** Circle(String name, **float** radius) {

**super**(name);

**this**.radius = radius;

}

@Override **public** **float** getArea() { // Override is for compiler to check for the override but this is not mandatory.

**return** 3.14f \*radius \* radius;

}

}

**package** overriding;

**public** **class** ShapeMain {

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

Shape shapes[] = **new** Shape[4];

shapes[0] = **new** Rectangle();

shapes[1] = **new** Circle("Shape 2", 5);

shapes[2] = **new** Rectangle("Shape 3", 2, 5);

shapes[3] = **new** Circle("Shape 4", 6);

System.***out***.println("Printing area of all 4 shapes");

**for** (Shape sh : shapes) {

**float** area = sh.getArea();

System.***out***.println(area);

}

}

}

**toString : Used existing class**

**package** containment;

**public** **class** Contact {

**private** String mobileNo, emailAddress;

**public** Contact() {

mobileNo = "45656564";

emailAddress = "sdsdf@gmail.com";

}

**public** Contact(String mobileNo, String emailAddress) {

**this**.mobileNo = mobileNo;

**this**.emailAddress = emailAddress;

}

**public** String getMobileNo() {

**return** mobileNo;

}

**public** **void** setMobileNo(String mobileNo) {

**this**.mobileNo = mobileNo;

}

**public** String getEmailAddress() {

**return** emailAddress;

}

**public** **void** setEmailAddress(String emailAddress) {

**this**.emailAddress = emailAddress;

}

@Override // toString overriding with Object class , if we will not have this method then the print method return in customer main class will return address.

To add this Source – Generate -- toString

**public** String toString() {

**return** "Contact [mobileNo=" + mobileNo + ", emailAddress=" + emailAddress + "]";

}

}

**package** containment;

**public** **class** CreditCard {

**private** String cardNo, bank;

**private** **int** creditLimit;

**public** CreditCard() {

cardNo = "46466";

bank = "HDFC";

creditLimit = 75000;

}

**public** CreditCard(String cardNo, String bank, **int** creditLimit) {

**this**.cardNo = cardNo;

**this**.bank = bank;

**this**.creditLimit = creditLimit;

}

**public** String getCardNo() {

**return** cardNo;

}

**public** **void** setCardNo(String cardNo) {

**this**.cardNo = cardNo;

}

**public** String getBank() {

**return** bank;

}

**public** **void** setBank(String bank) {

**this**.bank = bank;

}

**public** **int** getCreditLimit() {

**return** creditLimit;

}

**public** **void** setCreditLimit(**int** creditLimit) {

**this**.creditLimit = creditLimit;

}

}

**package** containment;

**public** **class** Customer {

**private** **int** customerId;

**private** String name;

**private** Contact contactDetails; // object of another class

**private** CreditCard cardDetails; // object of another class

**public** Customer() {

customerId = 1001;

name = "Dipti";

contactDetails = **new** Contact();

}

**public** Customer(**int** customerId, String name, Contact contactDetails, CreditCard cardDetails) {

**this**.customerId = customerId;

**this**.name = name;

**this**.contactDetails = contactDetails;

**this**.cardDetails = cardDetails;

}

**public** **int** getCustomerId() {

**return** customerId;

}

**public** **void** setCustomerId(**int** customerId) {

**this**.customerId = customerId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Contact getContactDetails() {

**return** contactDetails;

}

**public** **void** setContactDetails(Contact contactDetails) {

**this**.contactDetails = contactDetails;

}

**public** CreditCard getCardDetails() {

**return** cardDetails;

}

**public** **void** setCardDetails(CreditCard cardDetails) {

**this**.cardDetails = cardDetails;

}

}

**package** containment;

**public** **class** CustomerMain {

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

// creating customer without credit card.

Customer c1 = **new** Customer();

// Creating a customer with credit card.

Contact contact2 = **new** Contact("4535345", "ere@gmail.com");

CreditCard creditcard2 = **new** CreditCard("5453", "HSBC", 60000);

System.***out***.println(contact2); // this is implicit call, for explicit call System.***out***.println(contact2.toString());

Customer c2 = **new** Customer(1002, "Bill", contact2, creditcard2);

Contact ct = c1.getContactDetails();

System.***out***.println(ct.getEmailAddress());

String ct2 = c1.getContactDetails().getEmailAddress();

System.***out***.println(ct2);

System.***out***.println(c2.getContactDetails().getEmailAddress());

CreditCard cc = c1.getCardDetails();

**if** (cc != **null**)

System.***out***.println(cc.getCreditLimit());

**else**

System.***out***.println("No Credit card");

}

}

**Inherited toString:**

**package** Inheritence;

**public** **class** Player {

**private** String name;

**private** **int** age;

**public** **int** x;

**public** Player() {

name = "Sania";

age = 50;

System.***out***.println("In Player Class");

}

**public** Player(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

**public** Player(**int** age, String name) {

**this**.age = age;

**this**.name = name;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Player [name=" + name + ", age=" + age + "]";

}

}

**package** Inheritence;

**public** **class** Player {

**private** String name;

**private** **int** age;

**public** **int** x;

**public** Player() {

name = "Sania";

age = 50;

System.***out***.println("In Player Class");

}

**public** Player(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

**public** Player(**int** age, String name) {

**this**.age = age;

**this**.name = name;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Player [name=" + name + ", age=" + age + "]";

}

}

**package** Inheritence;

**public** **class** PlayerMain {

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

//CricketPayer cp = new CricketPayer();

CricketPayer cp1 = **new** CricketPayer("Dhoni", 50, 34300);

System.***out***.println(cp1);

/\* System.out.println(cp.getName());

System.out.println(cp.getAge());

System.out.println(cp.getRuns());\*/

System.***out***.println(cp1.getName());

System.***out***.println(cp1.getAge());

System.***out***.println(cp1.getRuns());

}

}

Equals(Object)

**package** containment;

**public** **class** Contact {

**private** String mobileNo, emailAddress;

**public** Contact() {

mobileNo = "45656564";

emailAddress = "sdsdf@gmail.com";

}

**public** Contact(String mobileNo, String emailAddress) {

**this**.mobileNo = mobileNo;

**this**.emailAddress = emailAddress;

}

**public** String getMobileNo() {

**return** mobileNo;

}

**public** **void** setMobileNo(String mobileNo) {

**this**.mobileNo = mobileNo;

}

**public** String getEmailAddress() {

**return** emailAddress;

}

**public** **void** setEmailAddress(String emailAddress) {

**this**.emailAddress = emailAddress;

}

@Override

**public** String toString() {

**return** "Contact [mobileNo=" + mobileNo + ", emailAddress=" + emailAddress + "]";

}

@Override

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

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + ((emailAddress == **null**) ? 0 : emailAddress.hashCode());

result = prime \* result + ((mobileNo == **null**) ? 0 : mobileNo.hashCode());

**return** result;

}

@Override

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

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Contact other = (Contact) obj;

**if** (emailAddress == **null**) {

**if** (other.emailAddress != **null**)

**return** **false**;

} **else** **if** (!emailAddress.equals(other.emailAddress))

**return** **false**;

**if** (mobileNo == **null**) {

**if** (other.mobileNo != **null**)

**return** **false**;

} **else** **if** (!mobileNo.equals(other.mobileNo))

**return** **false**;

**return** **true**;

}

}

**package** containment;

**public** **class** CreditCard {

**private** String cardNo, bank;

**private** **int** creditLimit;

**public** CreditCard() {

cardNo = "46466";

bank = "HDFC";

creditLimit = 75000;

}

**public** CreditCard(String cardNo, String bank, **int** creditLimit) {

**this**.cardNo = cardNo;

**this**.bank = bank;

**this**.creditLimit = creditLimit;

}

**public** String getCardNo() {

**return** cardNo;

}

**public** **void** setCardNo(String cardNo) {

**this**.cardNo = cardNo;

}

**public** String getBank() {

**return** bank;

}

**public** **void** setBank(String bank) {

**this**.bank = bank;

}

**public** **int** getCreditLimit() {

**return** creditLimit;

}

**public** **void** setCreditLimit(**int** creditLimit) {

**this**.creditLimit = creditLimit;

}

}

**package** containment;

**public** **class** Customer {

**private** **int** customerId;

**private** String name;

**private** Contact contactDetails; // object of another class

**private** CreditCard cardDetails; // object of another class

**public** Customer() {

customerId = 1001;

name = "Dipti";

contactDetails = **new** Contact();

}

**public** Customer(**int** customerId, String name, Contact contactDetails, CreditCard cardDetails) {

**this**.customerId = customerId;

**this**.name = name;

**this**.contactDetails = contactDetails;

**this**.cardDetails = cardDetails;

}

**public** **int** getCustomerId() {

**return** customerId;

}

**public** **void** setCustomerId(**int** customerId) {

**this**.customerId = customerId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Contact getContactDetails() {

**return** contactDetails;

}

**public** **void** setContactDetails(Contact contactDetails) {

**this**.contactDetails = contactDetails;

}

**public** CreditCard getCardDetails() {

**return** cardDetails;

}

**public** **void** setCardDetails(CreditCard cardDetails) {

**this**.cardDetails = cardDetails;

}

}

**package** containment;

**public** **class** CustomerMain {

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

// creating customer without credit card.

Customer c1 = **new** Customer();

// Creating a customer with credit card.

Contact contact2 = **new** Contact("4535345", "ere@gmail.com");

Contact contact3 = **new** Contact("4535345", "ere@gmail.com");

System.***out***.println(contact2.equals(contact3));

CreditCard creditcard2 = **new** CreditCard("5453", "HSBC", 60000);

//System.out.println(contact2);

Customer c2 = **new** Customer(1002, "Bill", contact2, creditcard2);

Contact ct = c1.getContactDetails();

System.***out***.println(ct.getEmailAddress());

String ct2 = c1.getContactDetails().getEmailAddress();

System.***out***.println(ct2);

System.***out***.println(c2.getContactDetails().getEmailAddress());

CreditCard cc = c1.getCardDetails();

**if** (cc != **null**)

System.***out***.println(cc.getCreditLimit());

**else**

System.***out***.println("No Credit card");

}

}

Abstract Class:

**package** abstract\_classes;

**public** **abstract** **class** MusicalInstrument {

**private** String id;

**private** **int** cost;

**public** **abstract** **void** play();

**public** MusicalInstrument() {

id = "m1";

cost = 15000;

}

**public** MusicalInstrument(String id, **int** cost) {

**this**.id = id;

**this**.cost = cost;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** **int** getCost() {

**return** cost;

}

**public** **void** setCost(**int** cost) {

**this**.cost = cost;

}

}

**package** abstract\_classes;

**public** **class** Piano **extends** MusicalInstrument {

**private** **int** strings;

**public** Piano() {

strings = 48;

}

**public** Piano(String id, **int** cost, **int** strings) {

**super**(id, cost);

**this**.strings = strings;

}

@Override

**public** **void** play() {

System.***out***.println("playing Piano with " + strings + " strings");

}

}

**package** abstract\_classes;

**public** **class** Tabla **extends** MusicalInstrument {

**private** String surface;

**public** Tabla() {

surface = "Hard";

}

**public** Tabla(String id, **int** cost, String surface) {

**super**(id, cost);

**this**.surface = surface;

}

**public** String getSurface() {

**return** surface;

}

**public** **void** setSurface(String surface) {

**this**.surface = surface;

}

@Override

**public** **void** play() {

System.***out***.println("Playing a table with: " +surface + "surface.");

}

}

**package** abstract\_classes;

**public** **class** MusicalInstrumentMain {

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

MusicalInstrument instruments[] = **new** MusicalInstrument[2];

instruments[0] = **new** Piano();

instruments[1] = **new** Tabla("M2", 7500, "Medium");

**for** (MusicalInstrument ins : instruments)

ins.play();

}

}

**Wrapper/ Argument:**

**public** **class** misc {

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

//program to accept information about a Person.

//as Name , age, weight in the form of command line argument

String name = args[0];

String sAge = args[1];

String sWeight = args[2];

**int** age = Integer.*parseInt*(sAge); // Integer wrapper

**float** weight = Float.*parseFloat*(sWeight);

StringBuffer personData = **new** StringBuffer();

personData.append("Name: ");

personData.append(name);

personData.append("Age: ");

personData.append(age);

personData.append("Weight: ");

personData.append(weight);

System.***out***.println(personData);

}

}

**To run above code , run configuration , select your class in main and then pass arguments. After this apply and run.**

**Interface :**

**package** interfaces;

**public** **interface** GreetingService {

String greet(String userName);

}

**package** interfaces;

**public** **class** HelloService **implements** GreetingService {

@Override

**public** String greet(String user) {

String message = "Hello " + user;

**return** message;

}

}

**package** interfaces;

**public** **class** WelcomeService **implements** GreetingService {

@Override

**public** String greet(String user) {

String message = "Welcome " + user;

**return** message;

}

}

**package** interfaces;

**public** **class** GreetingServiceMain {

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

String result;

GreetingService gs;

gs = **new** HelloService();

result = gs.greet("Ram");

System.***out***.println(result);

gs = **new** WelcomeService();

result = gs.greet("Shyam");

System.***out***.println(result);

}

}

**Interface example 2 :**

**package** interfaces;

**public** **interface** BisleriBottle {

**int** getVolume();

**int** getCost();

}

**package** interfaces;

**public** **class** BisleriPack {

**private** BisleriBottle bottleType;

**private** **int** quantity;

// Constructors

**public** BisleriPack() {

bottleType = **new** SmallBottle();

quantity = 20;

}

**public** BisleriPack(BisleriBottle bottleType, **int** quantity) {

**this**.bottleType = bottleType;

**this**.quantity = quantity;

}

// Getter - Setter

**public** BisleriBottle getBottleType() {

**return** bottleType;

}

**public** **void** setBottleType(BisleriBottle bottleType) {

**this**.bottleType = bottleType;

}

**public** **int** getQuantity() {

**return** quantity;

}

**public** **void** setQuantity(**int** quantity) {

**this**.quantity = quantity;

}

// Methods

**public** **int** getPackVolume() {

**return** quantity \* bottleType.getVolume();

}

**public** **int** getPackCost() {

**return** quantity \* bottleType.getCost();

}

}

**package** interfaces;

**public** **class** SmallBottle **implements** BisleriBottle {

@Override

**public** **int** getVolume() {

**return** 200;

}

@Override

**public** **int** getCost() {

**return** 5;

}

}

**package** interfaces;

**public** **class** MediumBottle **implements** BisleriBottle {

@Override

**public** **int** getVolume() {

**return** 300;

}

@Override

**public** **int** getCost() {

**return** 8;

}

}

**package** interfaces;

**public** **class** LargeBottle **implements** BisleriBottle {

@Override

**public** **int** getVolume() {

**return** 500;

}

@Override

**public** **int** getCost() {

**return** 10;

}

}

**package** interfaces;

**public** **class** BisleriMain {

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

BisleriPack packs[] = **new** BisleriPack[3];

packs[0] = **new** BisleriPack();

/\* BisleriBottle medium = new MediumBottle();

packs[1] = new BisleriPack(medium, 25);

\*/

BisleriBottle medium = **new** MediumBottle();

packs[1] = **new** BisleriPack(**new** BisleriBottle() {

@Override

**public** **int** getVolume() {

// **TODO** Auto-generated method stub

**return** 0;

}

@Override

**public** **int** getCost() {

// **TODO** Auto-generated method stub

**return** 0;

}

},25);

BisleriBottle large = **new** LargeBottle();

packs[2] = **new** BisleriPack(large, 15);

**int** totalVolume = 0 , totalCost = 0;

**for**(BisleriPack pack : packs) {

totalVolume += pack.getPackVolume();

totalCost += pack.getPackCost();

System.***out***.println(totalVolume);

System.***out***.println(totalCost);

}

}

}

**Design Patter : Singleton**

**package** patterns;

**public** **class** Singleton {

**private** **static** Singleton *ref*;

**private** Singleton() {

}

**static** **public** Singleton createObject() {

**if**(*ref* == **null**)

*ref* = **new** Singleton();

**return** *ref*;

}

}

//Singleton s1 = Singleton.createobject();

//Singleton s2 = Singleton.createobject();

//s1 = s2 = null;

//Singleton s3 = Singleton.createobject();

Error Handling: ArrayIndexOutOfBoundsException, NumberFormatException

**package** StringBuffer;

**public** **class** misc {

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

//program to accept information about a Person.

//as Name , age, weight in the form of command line argument

**try** {

String name = args[0];

String sAge = args[1];

String sWeight = args[2];

**int** age = Integer.*parseInt*(sAge);

**float** weight = Float.*parseFloat*(sWeight);

StringBuffer personData = **new** StringBuffer();

personData.append("Name: ");

personData.append(name);

personData.append("Age: ");

personData.append(age);

personData.append("Weight: ");

personData.append(weight);

System.***out***.println(personData);

}

**catch**(ArrayIndexOutOfBoundsException ex){

System.***out***.println("Enter atleast 3 values");

}

**catch**(NumberFormatException ex) {

System.***out***.println("age and weight should be numeric.");

}

}

}

**Error Handling: one Catch block handling multiple exceptions.**

**package** StringBuffer;

**public** **class** misc {

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

//program to accept information about a Person.

//as Name , age, weight in the form of command line argument

**try** {

String name = args[0];

String sAge = args[1];

String sWeight = args[2];

**int** age = Integer.*parseInt*(sAge);

**float** weight = Float.*parseFloat*(sWeight);

StringBuffer personData = **new** StringBuffer();

personData.append("Name: ");

personData.append(name);

personData.append("Age: ");

personData.append(age);

personData.append("Weight: ");

personData.append(weight);

System.***out***.println(personData);

}

**catch**(ArrayIndexOutOfBoundsException | NumberFormatException ex){ // pipe symbol is a bitwise operator

**if**(ex **instanceof** ArrayIndexOutOfBoundsException)

{

System.***out***.println("Enter atleast 3 values.");

}

**else** **if**(ex **instanceof** NumberFormatException)

{

System.***out***.println("The age or weight is invalid.");

}

}

}

}

**Cascading of exceptions:**

**package** StringBuffer;

**public** **class** misc {

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

//program to accept information about a Person.

//as Name , age, weight in the form of command line argument

**try** {

String name = args[0];

String sAge = args[1];

String sWeight = args[2];

**int** age = Integer.*parseInt*(sAge);

**float** weight = Float.*parseFloat*(sWeight);

StringBuffer personData = **new** StringBuffer();

personData.append("Name: ");

personData.append(name);

personData.append("Age: ");

personData.append(age);

personData.append("Weight: ");

personData.append(weight);

System.***out***.println(personData);

Object obj = personData;

String dataAboutPerson = (String)obj; // Dynamic time of obj is StringBuffer which fails during run time.

System.***out***.println(dataAboutPerson);

}

**catch**(NullPointerException ex) {

System.***out***.println("Null Pointer Exception");

}

**catch**(ArrayIndexOutOfBoundsException | NumberFormatException ex){ // pipe symbol is a bitwise operator

**if**(ex **instanceof** ArrayIndexOutOfBoundsException)

{

System.***out***.println("Enter atleast 3 values.");

}

**else** **if**(ex **instanceof** NumberFormatException)

{

System.***out***.println("The age or weight is invalid.");

}

}

**catch** (Exception ex) {

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

}

**catch** (Throwable ex) {

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

}

}

}

Exception: Throws

**package** Exceptions;

**public** **class** ThrowsMain {

**public** **static** **void** method3() **throws** Exception {

System.***out***.println("In method3");

}

**public** **static** **void** method2() **throws** Exception {

*method3*();

}

**public** **static** **void** method1() **throws** Exception {

*method2*();

}

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

**try** {

*method1*();

} **catch** (Exception e) {

e.printStackTrace(); // It is helpful in debugging the code.

}

}

}

**Another example with unchecked exception**

**package** Exceptions;

**public** **class** ThrowsMain {

**public** **static** **void** method3() **throws** Exception {

System.***out***.println("In method3");

}

**public** **static** **void** method2() **throws** Exception {

*method3*();

}

**public** **static** **void** method1() **throws** Exception {

*method2*();

}

**public** **static** **void** method4() **throws** RuntimeException {

}

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

*method4*(); // Throws in main method or try catch block not mandate for method4 as it is unchecked exception.

**try** {

*method1*();

} **catch** (Exception e) {

e.printStackTrace(); // It is helpful in debugging the code.

}

}

}

**Problem Statement**

**Candidate : name , email address , age (25 to 60)**

**Runtime / Checked**

**InvalidAgeException extends Exception**

**InvalidEmailException extends Exception**

**package** Exceptions;

**public** **class** InvalidAgeException **extends** Exception {

**private** **int** invalidAge;

**public** InvalidAgeException(String errorMessage, **int** invalidAge) {

**super**(errorMessage);

**this**.invalidAge = invalidAge;

}

@Override

**public** String getMessage() {

String msg = **super**.getMessage();

**return** msg + ": " + invalidAge;

}

}

**package** Exceptions;

**public** **class** InvalidEmailException **extends** Exception {

**public** String invalidEmail;

**public** InvalidEmailException(String errormessage, String invalidEmail) {

**super**(errormessage);

**this**.invalidEmail = invalidEmail;

}

@Override

**public** String getMessage() {

String msg = **super**.getMessage();

**return** msg + ": " + invalidEmail;

}

}

**package** Exceptions;

**import** java.util.regex.Pattern;

**public** **class** Candidate {

**private** String name, emailAddress;

**private** **int** age;

**public** Candidate() {

name = "James Gosling";

emailAddress = "james.gosling@oracle.com";

age =59;

}

**public** Candidate(String name, String emailAddress, **int** age) **throws** InvalidAgeException, InvalidEmailException{

**this**.name = name;

**boolean** isAt = emailAddress.contains("@");

**boolean** isDot = emailAddress.contains(".");

**if**(!isAt || !isDot) {

InvalidEmailException ie = **new** InvalidEmailException("Email address is invalid.", emailAddress);

**throw** ie;

}

**this**.emailAddress = emailAddress;

**if**(age < 25 || age >60) {

InvalidAgeException ie = **new** InvalidAgeException("Age is invalid", age);

**throw** ie;

}

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getEmailAddress() {

**return** emailAddress;

}

**public** **void** setEmailAddress(String emailAddress) **throws** InvalidEmailException{

**boolean** isAt = emailAddress.contains("@");

**boolean** isDot = emailAddress.contains(".");

**if**(!isAt || !isDot) {

InvalidEmailException ie = **new** InvalidEmailException("Email address is invalid.", emailAddress);

**throw** ie;

}

**this**.emailAddress = emailAddress;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) **throws** InvalidAgeException {

**if**(age < 25 || age >60) {

InvalidAgeException ie = **new** InvalidAgeException("Age is invalid", age);

**throw** ie;

}

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Candidate [name=" + name + ", emailAddress=" + emailAddress + ", age=" + age + "]";

}

}

**package** Exceptions;

**public** **class** CandidateMain {

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

Candidate c1 = **null**, c2 = **null**;

c1 = **new** Candidate();

**try** {

c2 = **new** Candidate("Dipti", "dipti.jain600@gmail.com", 30);

} **catch** (InvalidAgeException | InvalidEmailException e) {

String errMsg = e.getMessage();

System.***out***.println(errMsg);

}

**finally** {

System.***out***.println(c1);

**if**(c2 !=**null**)

System.***out***.println(c2);

}

}

}

**Rethrowing an exception:**

**Lambda Expression :**

**package** interfaces;

@FunctionalInterface

**public** **interface** GreetingService {

String greet(String userName);

}

**package** interfaces;

**public** **class** GreetingServiceMain {

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

GreetingService service1 =

(user) -> "Hello" +user;

GreetingService service2 =

(user) -> "Welcome" +user;

String result = service1.greet("Ram");

System.***out***.println(result);

result = service2.greet("Shyam");

System.***out***.println(result);

/\* String result;

GreetingService gs;

gs = new HelloService();

result = gs.greet("Ram");

System.out.println(result);

gs = new WelcomeService();

result = gs.greet("Shyam");

System.out.println(result);\*/

}

}

**Multithread:**

**1. Extending thread class**

**package** multithreading;

**public** **class** ThreadImp **extends** Thread {

**private** String message;

**private** **int** delayTime;

**public** ThreadImp(String message, **int** delayTime) {

**this**.message = message;

**this**.delayTime = delayTime;

}

**public** **void** run() { // as soon as execution begin it will call run method.

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

System.***out***.println(message);

**try** {

Thread.*sleep*(delayTime);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

}

**package** multithreading;

**public** **class** ThreadMain {

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

Thread t1 = **new** ThreadImp("Hello", 1000); // BORN

Thread t2 = **new** ThreadImp("Welcome", 500);

t1.start(); // Ready t1, if we change t1.start to t1.run then it will behave like main thread only , this will print all hello first and then welcome.

t2.start();

}

}

1. **Implementing runnable interface**

**package** multithreading;

**public** **class** RunnableImpl **implements** Runnable {

**private** String message;

**private** **int** delayTime;

**public** RunnableImpl(String message, **int** delayTime) {

**this**.message = message;

**this**.delayTime = delayTime;

}

@Override

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

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

System.***out***.println(message);

**try** {

Thread.*sleep*(delayTime);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

}

**package** multithreading;

**public** **class** RunnableMain{

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

Runnable r1 = **new** RunnableImpl("Hello", 1000);

Runnable r2 = **new** RunnableImpl("Welcome", 500);

Thread t1 = **new** Thread(r1);

Thread t2 = **new** Thread(r2);

t1.start();

t2.start();

}

}

**Join method example :**

**package** multithreading;

**public** **class** RunnableMain{

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

Runnable r1 = **new** RunnableImpl("Hello", 1000);

Runnable r2 = **new** RunnableImpl("Welcome", 500);

Thread t1 = **new** Thread(r1);

Thread t2 = **new** Thread(r2);

t1.start();

t2.start();

**try** {

t2.join();

}

**catch**(Exception e) {

e.printStackTrace();

}

System.***out***.println("Message"); // message will print after threads t2 is dead.

}

}

**package** multithreading;

**public** **class** RunnableImpl **implements** Runnable {

**private** String message;

**private** **int** delayTime;

**public** RunnableImpl(String message, **int** delayTime) {

**this**.message = message;

**this**.delayTime = delayTime;

}

@Override

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

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

System.***out***.println(message);

**try** {

Thread.*sleep*(delayTime);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

}

**Multi- threading Synchronization:**

**package** multithreading;

**public** **class** Message {

**private** String greetingMessage;

**public** Message(String greetingMessage) {

**super**();

**this**.greetingMessage = greetingMessage;

}

**Public synchronized** **void** printMessage() **throws** Exception{ // if synchronized is removed from here then it will be Asynchronized.

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

Thread.*sleep*(1000);

System.***out***.println(greetingMessage);

Thread.*sleep*(1000);

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

Thread.*sleep*(1000);

}

}

**package** multithreading;

**public** **class** MessageThread **extends** Thread {

**private** Message messageObject; // Message class object

**public** MessageThread(Message messageObject) {

**super**();

**this**.messageObject = messageObject;

}

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

**try** {

messageObject.printMessage();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**package** multithreading;

**public** **class** MessageMain {

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

Message m1 = **new** Message("Welcome");

Thread t1 = **new** MessageThread(m1);

Thread t2 = **new** MessageThread(m1);

t1.start();

t2.start();

}

}

**Synchronized Block :**

**package** multithreading;

**public** **class** Message {

**private** String greetingMessage;

**public** Message(String greetingMessage) {

**super**();

**this**.greetingMessage = greetingMessage;

}

**public** **void** printMessage() **throws** Exception{

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

Thread.sleep(1000);

System.out.println(greetingMessage);

Thread.sleep(500);

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

Thread.sleep(1000);

}

}

**package** multithreading;

**public** **class** MessageThread **extends** Thread {

**private** Message messageObject; // Message class object

**public** MessageThread(Message messageObject) {

**super**();

**this**.messageObject = messageObject;

}

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

**synchronized** (messageObject) {

**try** {

messageObject.printMessage();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

}

**package** multithreading;

**public** **class** MessageMain {

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

Message m1 = **new** Message("Welcome");

Thread t1 = **new** MessageThread(m1);

Thread t2 = **new** MessageThread(m1);

t1.start();

t2.start();

}

}

**IO Programming :**

**package** io\_programming;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.io.InputStream;

**public** **class** FileReadMain {

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

InputStream in = **null**;

String filename = "C:\\Users\\adm\\Documents\\DJ57741\\greeting.txt";

**try** {

in = **new** FileInputStream(filename);

**while**(**true**)

{

**int** data = in.read(); // read method throws IOException so had to add try catch block.

**if**(data == -1) {

**break**;

}

System.***out***.print((**char**)data);

}

} **catch** (FileNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} // InputStream is an abstract class so cannot create object so will create ojc/ref of super type.

**catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**finally** {

**try** {

in.close();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

}

**Try with resources : so no need to close the resources.**

**package io\_programming;**

**import java.io.FileInputStream;**

**import java.io.FileNotFoundException;**

**import java.io.IOException;**

**import java.io.InputStream;**

**public class FileReadMain {**

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

**String filename = "C:\\Users\\adm\\Documents\\DJ57741\\greeting.txt";**

**try (InputStream in = new FileInputStream(filename)){**

**while(true)**

**{**

**int data = in.read(); // read method throws IOException so had to add try catch block.**

**if(data == -1) {**

**break;**

**}**

**System.out.print((char)data);**

**}**

**} catch (FileNotFoundException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**} // InputStream is an abstract class so cannot create object so will create ojc/ref of super type.**

**catch (IOException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}**

**}**

**Buffered Input Stream :**

**package** io\_programming;

**import** java.io.BufferedInputStream;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.io.InputStream;

**public** **class** FileReadMain {

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

String filename = "C:\\Users\\adm\\Documents\\DJ57741\\greeting.txt";

**try** (InputStream in = **new** FileInputStream(filename);

InputStream bin = **new** BufferedInputStream(in)){; // if we want to pass size then (in, size(bytes))

**while**(**true**)

{

**int** data = bin.read(); // read method throws IOException so had to add try catch block.

**if**(data == -1) {

**break**;

}

System.***out***.print((**char**)data);

}

} **catch** (FileNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} // InputStream is an abstract class so cannot create object so will create ojc/ref of super type.

**catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**Write into file :**

**package** io\_programming;

**import** java.io.BufferedOutputStream;

**import** java.io.FileOutputStream;

**import** java.io.OutputStream;

**public** **class** FileWriteMain {

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

String java = "Java is platform independent language.";

**try**(OutputStream fout = **new** FileOutputStream("java.txt");

OutputStream bout= **new** BufferedOutputStream(fout)){

**byte** data[] = java.getBytes();

bout.write(data); // write first to buffer and when buffered is closed it will write to main file.

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

**Write File with append data**

**package** io\_programming;

**import** java.io.BufferedOutputStream;

**import** java.io.FileOutputStream;

**import** java.io.OutputStream;

**public** **class** FileWriteMain {

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

String java = "Java is platform independent language. - changed";

**try**(OutputStream fout = **new** FileOutputStream("java.txt", **true**); // true appended data here.

OutputStream bout= **new** BufferedOutputStream(fout)){

**byte** data[] = java.getBytes();

bout.write(data); // write first to buffer and when buffered is closed it will write to main file.

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

**Random Access File :**

**package** io\_programming;

**import** java.io.RandomAccessFile;

**public** **class** RandomAccessFileMain {

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

String filename = "greeting.txt";

**try**(RandomAccessFile rf = **new** RandomAccessFile(filename, "r")){ // r is for read and w is for right.

**long** size = rf.length();

**long** position = size/2;

rf.seek(size - position);

**byte** data[] = **new** **byte**[(**int**) position];

rf.read(data);

String contents = **new** String (data);

System.***out***.println(contents);

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

**Buffered Reader:**

**package** io\_programming;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.Reader;

**public** **class** BufferReaderMain {

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

String filename = "java.txt";

File currentFile = **new** File(filename);

**if**(currentFile.exists()) {

**try**(Reader fr = **new** FileReader(filename);

BufferedReader br = **new** BufferedReader(fr)){

**while**(**true**) {

String line = br.readLine();

**if**(line == **null**) {

**break**;

}

System.***out***.println(line);

}

}

**catch**(Exception ex) {

ex.printStackTrace();

}

}

**else** {

System.***out***.println("File does not exists.");

}

}

}

**Scanner class :**

**package** io\_programming;

**import** java.util.Scanner;

**public** **class** UserInputMain {

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

**try**(Scanner scr = **new** Scanner(System.***in***)){

System.***out***.println("Enter employee number");

**int** empno = scr.nextInt();

scr.nextLine();

System.***out***.println("Enter name: ");

String empName = scr.nextLine();

System.***out***.println("Emp No: " +empno);

System.***out***.println("Name: " +empName );

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

**Another way to write above program :**

**package** io\_programming;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.util.Scanner;

**public** **class** UserInputMain {

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

InputStream is = System.***in***;

**try**( Scanner scr = **new** Scanner(is)){

System.***out***.println("Enter employee number");

**int** empno = scr.nextInt();

scr.nextLine();

System.***out***.println("Enter name: ");

String empName = scr.nextLine();

System.***out***.println("Enter Salary: ");

**float** sal = scr.nextFloat();

System.***out***.println("Emp No: " +empno);

System.***out***.println("Name: " +empName );

System.***out***.println("Salary: " +sal );

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

Serializable :

**package** io\_programming;

**import** java.io.Serializable;

**public** **class** Doctor **implements** Serializable {

// Method will not define thT'S WHY we will call it as markup interface.

**private** String name, specialization;

**public** Doctor() {

**this**.name = "Mr. Tushar";

**this**.specialization = "Dentist";

}

**public** Doctor(String name, String specialization) {

**this**.name = name;

**this**.specialization = specialization;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getSpecialization() {

**return** specialization;

}

**public** **void** setSpecialization(String specialization) {

**this**.specialization = specialization;

}

@Override

**public** String toString() {

**return** "Doctor [name=" + name + ", specialization=" + specialization + "]";

}

}

**package** io\_programming;

**import** java.io.FileOutputStream;

**import** java.io.ObjectOutputStream;

**import** java.io.OutputStream;

**public** **class** SerializationMain {

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

String filename = "Doctor.txt";

**try**(OutputStream fout = **new** FileOutputStream(filename);

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

Doctor doc = **new** Doctor();

out.writeObject(doc);

System.***out***.println("Doctor object serialized");

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

**package** io\_programming;

**import** java.io.FileInputStream;

**import** java.io.InputStream;

**import** java.io.ObjectInputStream;

**public** **class** DeserializationMain {

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

String filename = "Doctor.txt";

**try**(InputStream is = **new** FileInputStream(filename);

ObjectInputStream oi = **new** ObjectInputStream(is)){

Doctor doc = **new** Doctor();

Object obj = oi.readObject();

System.***out***.println(obj);

}

**catch** (Exception e) {

e.printStackTrace();

}

}

}