**Abstract # 1**

**1.** Create an abstract class Student [Fields: name; Method: constructor(), getName(), abstract getRemainingHours(), toString()].

**2.** Create a CompSciStudent class that extends Student class. [Fields: CS\_HOURS (40), csHoursTaken; method setCsHoursTaken, toString (call superclass toString + subclass)]

abstract class Student

{

private String name;

public Student(String name){this.name = name;}

public String getName(){return this.name;}

public abstract int getRemainingHours();

public String toString()

{

String st = "Name: " + this.name;

return st;

}

}

class CompSciStudent extends Student

{

private final int CS\_HOURS = 40;

private int csHoursTaken;

public CompSciStudent(String name){super(name);}

public void setCsHoursTaken(int csHoursTaken){this.csHoursTaken = csHoursTaken;}

public int getRemainingHours(){return CS\_HOURS - csHoursTaken;}

public String toString()

{

String st = super.toString() + "\tRemaining CS hours: " + this.getRemainingHours();

return st;

}

}

public class Abstract1

{

public static void main(String[] args)

{

CompSciStudent student = new CompSciStudent("Name1");

student.setCsHoursTaken(1);

System.out.println(student);

}

}

**Aggregation # 1**

**1.** Create class Instructor [Fields: name, officeNum; Methods: constructor, toString]

**2.** Create class Book [Fields: title, author; Methods: constructor, toString]

**3.** Create class Course [Fields: courseName, Instructor, Book; Methods: getCourseName, getInstructor, getBook, toString]

**4.** Create a Display the complete Course information. [Has-a relationship, the course has a book and instructor.]

class Instructor

{

private String name;

private String officeNum;

public Instructor(String name, String officeNum)

{

this.name = name;

this.officeNum = officeNum;

}

public Instructor(Instructor instructor){this(instructor.name, instructor.officeNum);}

public String toString()

{

String st = "\nInstructor's information: " + "\nName: " + this.name + "\nOffice Number: " + this.officeNum;

return st;

}

}

class Book

{

private String title;

private String author;

public Book(String title, String author)

{

this.title = title;

this.author = author;

}

public Book(Book book){this(book.title, book.author);}

public String toString()

{

String st = "\nBook Information: " + "\nTitle: " + this.title + "\nAuthor " + this.author;

return st;

}

}

class Course

{

private String courseName;

private Instructor instructor;

private Book book;

public Course(String courseName, Instructor instructor, Book book)

{

this.courseName = courseName;

this.instructor = new Instructor(instructor);

this.book = new Book(book);

}

public String getCourseName(){return courseName;}

public Instructor getInstructor(){return new Instructor(instructor);}

public Book getBook(){return new Book(book);}

public String toString()

{

String st = "Course Information: " + "\nCourse: " + courseName + book + instructor;

return st;

}

}

class Aggregation1

{

public static void main(String[] args)

{

Instructor instructor = new Instructor("Ira N. Levine", "3125B");

Book book = new Book("General Chemistry", "Ira N. Levine");

Course course = new Course("Chemistry 1", instructor, book);

System.out.println(course);

}

}

**Aggregation # 2**

**1.** Create a class Stock for a Stock market. [Fields: symbol, sharePrice; Methods: constructor, def. constructor("-", 0.0), get/set methods for symbol/sharePrice, toString]

**2.** Write a method createStock() to create a Stock object. Write a method displayStock() to display the state of the object.

**3.** Write a method equals that compares the two companies. Write a method copyStock() to copy Stock object that accepts Object. [Test it with equals and == method]

**4.** Write a getObjectCounter() counting objects created. Create a class StockPurchase that represents a stock purchase [Fields: stoc, numShares; Methods: getCost, toString()].

import java.text.DecimalFormat;

class Stock

{

private String symbol;

private double sharePrice;

private int objectCounter = 0;

public Stock(String symbol, double sharePrice)

{

this.symbol = symbol;

this.sharePrice = sharePrice;

objectCounter++;

}

public Stock()

{

this.symbol = "-";

this.sharePrice = 0.0;

objectCounter++;

}

public Stock(Stock stock) // copy constructor for the StockPurchase class.

{

this.symbol = stock.symbol;

this.sharePrice = stock.sharePrice;

}

public void setSymbol(String symbol){this.symbol = symbol;}

public void setSharePrice(double sharePrice){this.sharePrice = sharePrice;}

public String getSymbol(){return this.symbol;}

public double getSharePrice(){return this.sharePrice;}

public int getObjectCounter(){return objectCounter;}

public boolean equals(Object obj)

{

if(obj != null && obj instanceof Stock && this.getSymbol().equals(((Stock)obj).getSymbol()) && this.getSharePrice() == ((Stock)obj).getSharePrice())

return true;

return false;

}

public Stock copyObject(Stock stock){return new Stock(stock.symbol, stock.sharePrice);}

public Stock copyStock(){return new Stock(this.symbol, this.sharePrice);}

public String toString()

{

DecimalFormat formatter = new DecimalFormat("#,##0.00");

String st = "Company Symbol: " + this.symbol + "\tShare Price: $" + formatter.format(this.sharePrice);

return st;

}

}

class StockPurchase

{

private Stock stock;

private int numShares;

public StockPurchase(Stock stock, int numShares)

{

this.stock = new Stock(stock);

this.numShares = numShares;

}

public double getCost(){return stock.getSharePrice() \* this.numShares;}

public String toString()

{

DecimalFormat formatter = new DecimalFormat("#,##0.00");

String st = "Symbol: " + this.stock.getSymbol() + "\tShare Price: $" + formatter.format(this.stock.getSharePrice()) + "\tAmount : $" + formatter.format(this.getCost());

return st;

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Stock stock = createStock("XYZ", 20.0);

displayStock(stock);

StockPurchase stockPurchase = new StockPurchase(stock, 5);

System.out.println(stockPurchase);

Stock origStock = createStock("CompanyX", 100);

Stock copyStock = origStock.copyStock();

if(origStock == copyStock) System.out.println("The two stocks reference the same objects");

if(origStock.equals(copyStock)) System.out.println("The two stocks are equal meaningfully");

}

public static Stock createStock(String symbol, double sharePrice){ return new Stock(symbol, sharePrice);}

public static void displayStock(Stock stock) {System.out.println(stock);}

}

**Arrays**

**1.** Declare String array [size 2], no assignment. What will be printed?

**2.** Write methods [generic type] to display elements of an array using for loop and an enhanced for loop.

**3.** Write a method copyArr() to copy an array. Test String copy and original array.

**4.** Write a method compareArr() to compare arrays.

**5.** Write a method sumArr() to sum elements of 2D array. [display sum and average]

**6.** Write a method display2DArr() to display 2D array.

**7.** Write a method sum2DArr() to sum all elements of 2D array.

**8.** Write a method sumRows2DArr() to sum the rows of 2D array.

**9.** Write a method sumColumns2DArr() to sum the columns of 2D array.

**10.** Declare and print the number of columns and rows in a Ragged arrays.

public class ArrayListEx1

{

public static void main(String[] args)

{

String[] arr1 = new String[2];

displayArr(arr1);

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

int[] copyArr = copyArr(origArr);

if(origArr == copyArr)

System.out.println("References of two the same objects.");

else

System.out.println("References of two different objects.");

System.out.println(compareArr(origArr, copyArr)); // Strings are the same

System.out.println("Sum of elements is " + sumArr(origArr) + "\tAverage: " + sumArr(origArr) / origArr.length); // Sum of elements is 15.0

int[][] arr2 = {{3, 2}, {4, 2}};

display2DArr(arr2);

System.out.println("\nThe sum of 2D array is " + sum2DArr(arr2)); // The sum of 2D array is 6

sumRows2DArr(arr2);

sumColumns2DArr(arr2);

int[][] raggedArr = new int[2][];

raggedArr[0] = new int[2];

raggedArr[1] = new int[4];

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

System.out.println("Row " + i + " has " + raggedArr[i].length + " columns");

}

public static void sumColumns2DArr(int[][] arr)

{

for(int column = 0; column < arr[0].length; column++)

{

int sum = 0;

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

sum += arr[row][column];

System.out.println("The sum of column " + (column + 1) + " is " + sum);

}

}

public static void sumRows2DArr(int[][] arr)

{

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

{

int sum = 0;

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

sum += arr[row][column];

System.out.println("The sum of row " + (row + 1) + " is " + sum);

}

}

public static int sum2DArr(int[][] arr)

{

int sum = 0;

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

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

sum += arr[row][column];

return sum;

}

public static void display2DArr(int[][] arr)

{

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

{

System.out.println();

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

System.out.print(arr[row][column] + " ");

}

}

public static double sumArr(int[] arr)

{

int sum = 0;

for(int num : arr)

sum += num;

return sum;

}

public static String compareArr(int[] arr1, int[] arr2)

{

int count = 0;

if(arr1.length == arr2.length)

for(int i = 0; i < arr1.length; i++)

if(arr1[i] == arr2[i])

count++;

if(count == arr1.length)

return "Strings are the same";

return "Strings are not the same";

}

public static int[] copyArr(int[] arr)

{

int[] tempArr = new int[arr.length];

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

tempArr[i] = arr[i];

return tempArr;

}

public static <E> void displayArr(E[] arr)

{

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

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

for(E element : arr)

System.out.println(element);

}

}

**Collection # 1**

\* Create class Person[Fields: name, age; Methods: getName(), getAge(), toString()].

\* Override equals() method to compare the Person objects by their name and age.

\* Override the hashCode method based on the person's name.

\* Test hashCode() and equals() method with a few instances.

\* Create an ArrayList of Person instances and add 2 Person objects to the list.

\* Create an iterator for the list and print Person name and age using an iterator.

\* Get the index of any added person object.

\* Remove the second person from the list.

\* Convert the List to array. Display an array.

\* Create a HashSet to store Person objects.

\* Add two Person objects to the Set.

\* Display elements in the hashSet using an iterator object.

\* Search for any person object in the set.

import java.util.\*;

class Person

{

private int age;

private String name;

public Person(String name, int age)

{

this.age = age;

this.name = name;

}

public int getAge()

{

return this.age;

}

public String getName()

{

return name;

}

public int hashCode()

{

return name.hashCode();

}

public boolean equals(Object obj)

{

if(obj != null && obj instanceof Person && ((Person) obj).getAge() == this.age && ((Person) obj).getName().equals(this.name))

return true;

return false;

}

public String toString()

{

String st = "Name: " + this.name + "\tAge: " + this.age;

return st;

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Person person1 = new Person("Chris", 25);

Person person2 = new Person("Mary", 26);

if(person1.equals(person2))

System.out.println("Same Object"); // Different Object

else

System.out.println("Different Object");

if(person1.hashCode() == person2.hashCode()) // Good HashCode

System.out.println("Good HashCode");

else

System.out.println("Bad HashCode");

List<Person> myList = new ArrayList<Person>();

myList.add(new Person("Chris", 52));

myList.add(new Person("Ann", 42));

ListIterator<Person> iterator = myList.listIterator();

while(iterator.hasNext())

{

Person person = iterator.next();

System.out.println("Name: " + person.getName() + "\tAge: " + person.getAge());

}

System.out.println(myList.indexOf(new Person("Ann", 42)));

myList.remove(1);

Object[] objList = myList.toArray();

for(Object o : objList)

System.out.println("Name: " + ((Person) o).getName() + "\tAge: " + ((Person) o).getAge());

Set<Person> mySet = new HashSet<Person>();

mySet.add(new Person("Ann", 34));

mySet.add(new Person("Chris", 43));

Iterator<Person> iterator = mySet.iterator();

while(iterator.hasNext())

{

Person tempPerson = iterator.next();

System.out.println("Name: " + tempPerson.getName() + "\tAge: " + tempPerson.getAge());

}

if(mySet.contains(new Person("Ann", 34)))

System.out.println("The object is in the set"); // The object is in the set

}

}

**Collection # 2**

\* Use Person class, create a PersonComparator Class.

\* Add a compare() method that compares Person objects by their names and returns an integer value.

\* Create a TreeSet and pass an instance of PersonComparator to it.

\* Add three Person objects to the TreeSet: person1[Name: ccc Age: 55], person2[Name: aaa Age: 15], person3[Name: ccc Age: 35]

\* Display the elements in the TreeSet.

import java.util.\*;

import java.util.Comparator;

class PersonComparator <T extends Person> implements Comparator<T>

{

public int compare(T person1, T person2)

{

return (person1.getName().compareToIgnoreCase(person2.getName()));

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Set<Person> mySet = new TreeSet<Person>(new PersonComparator<Person>());

mySet.add(new Person("ccc", 55));

mySet.add(new Person("bbb", 15));

mySet.add(new Person("aaa", 35));

System.out.println(mySet);

// [Name: aaa Age: 35, Name: bbb Age: 15, Name: ccc Age: 55]

}

}

**Collection # 3**

/\*\*

\* Create an ArrayList of Person objects, add 3 objects to the list and display all the objects.

\* Display elements before/after sorting by age [Comparable interface, compareTo()] and by name [Comparator interface, compare()].

\*/

import java.util.\*;

class AgeComparator <T extends Person> implements Comparator<T>

{

public int compare(T person1, T person2)

{

return (person1.getName().compareToIgnoreCase(person2.getName()));

}

}

class Person implements Comparable<Person>

{

public int compareTo(Person p)

{

return (new Integer(getAge()).compareTo(new Integer(p.getAge())));

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

ArrayList<Person> nameList = new ArrayList<Person>();

nameList.add(new Person("Person3", 54));

nameList.add(new Person("Person1", 45));

nameList.add(new Person("Person2", 65));

System.out.println("Original: " + nameList);

Collections.sort(nameList);

System.out.println("Sorted by age: " + nameList);

Collections.sort(nameList, new AgeComparator<Person>());

System.out.println("Sorted by name: " + nameList);

}

}

**Collection # 4**

\* 1. Create an ArrayList to hold String Objects.

\* 2. Add names [Chris, Ann] to the ArrayList.

\* 3. Add the name James after Chris using a list iterator.

\* 4. Get another list iterator.

\* 5. Replace the first name with Mary using list iterator.

\* 6. Remove the last element using list iterator.

\* 7. Create a LinkedList to hold String objects.

\* 8. Add names [Chris, Ann] to the ArrayList.

\* 9. Get a list iterator to transverse the list.

\* 10. Display the names [for loop, enhanced for loop, list iterator].

\* 11. Display the names in the reverse order [for loop, list iterator].

\* 12. Display the size and elements [square brackets] of ArrayList.

\* 13. Create an ArrayList of Strings to hold some names.

\* 14. Add the following names to it [Catherine, Mary].

\* 15. Get the size and all elements[square brackets] of an ArrayList.

\* 16. Insert Chris into the first position of the list, and Kate into the last position of the list. Display the result.

\* 17. Replace the first position with "FirstName" and the last postion with "LastName" Disply the result.

import java.util.\*;

public class ArrayListEx3

{

public static void main(String[] args)

{

List<String> myList1 = new ArrayList<String>();

myList1.add("Chris");

myList1.add("Ann");

ListIterator<String> iterator1 = myList1.listIterator();

boolean done = false;

while(iterator1.hasNext() && done == false)

if(iterator1.next().compareToIgnoreCase("Chris") == 0)

{

iterator1.add("Jack");

done = true;

}

System.out.println(myList1); // [Chris, Jack, Ann]

ListIterator<String> iterator2 = myList1.listIterator();

if(iterator2.hasNext())

{

iterator2.next();

iterator2.set("Mary");

}

System.out.println(myList1); // [Mary, Jack, Ann]

while(iterator2.hasNext())

iterator2.next();

iterator2.remove();

System.out.println(myList1); // [Mary, Jack]

List<String> myList2 = new LinkedList<String>();

List<String> myList3 = new ArrayList<String>();

myList3.add("Chris");

myList3.add("Ann");

ListIterator<String> iterator = myList3.listIterator();

while(iterator.hasNext())

System.out.println(iterator.next()); // Chris, Ann

for(int i = 0; i < myList3.size(); i++) // Chris, Ann

System.out.println(myList3.get(i));

for(String st : myList3) // Chris, Ann

System.out.println(st);

while(iterator.hasPrevious()) // Ann, Chris

System.out.println(iterator.previous());

for(int i = myList3.size() - 1; i >= 0; i--) // Ann, Chris

System.out.println(myList3.get(i));

System.out.println(myList3.size() + " elements: " + myList3); // 2 elements: [Chris, Ann]

ArrayList<String> names = new ArrayList<String>();

names.add("Catherine");

names.add("Mary");

System.out.printf("There are %d elements in the ArrayList: %s\n", names.size() , names);

// There are 2 elements in the ArrayList: [Catherine, Mary]

names.add(0, "Chris");

names.add(names.size(), "Kate");

System.out.println(names); // [Chris, Catherine, Mary, Kate]

names.set(0, "FirstName");

names.set(names.size() - 1, "LastName");

System.out.println(names); // [FirstName, Catherine, Mary, LastName]

}

}

**Collection # 5**

\* 1. Create a HashSet to hold String objects.

\* 2. Add some names [Chris, Ann] to the ArrayList.

\* 3. Display the elements of the set in Square brackets.

\* 4. Display elements using an enhanced for loop.

\* 5. Add a duplicate name "Chris" to the set using an if-stmt and add method.

\* 6. Get an iterator for the set.

\* 7. Display the elements in the set using an iterator.

\* 8. Search for "Chris" in the set with an if-stmt.

\* 9. Create a LinkedHashSet to hold String objects, create a HashSet to hold String objects..

\* 10. Add some names [Kate, Mike] to the HashSet, add some names [Mike, Kate] to the LinkedHashSet.

\* 11. Display elements of the LinkedHashSet, Display elements of the HashSet.

\* 12. Create a TreeSet to hold String objects.

\* 13. Add some names [Chris, Ann] to the TreeSet.

\* 14. Display elements of the TreeSet.

import java.util.\*;

public class ArrayListEx1

{

public static void main(String[] args)

{

Set<String> mySet = new HashSet<String>();

mySet.add("Chris");

mySet.add("Ann");

System.out.println(mySet); // [Chris, Ann]

for(String s : mySet)

System.out.println(s); // Chris, Ann

mySet.add("Mary");

if(!mySet.add("Chris"))

System.out.println("Chris was not added again."); // Chris was not added again.

System.out.println(mySet); // [Mary, Chris, Ann]

Iterator<String> iterator = mySet.iterator();

while(iterator.hasNext())

System.out.println(iterator.next()); // Mary, Chris. Ann

if(mySet.contains("Chris"))

System.out.println("Chris is found"); // Chris is found

Set<String> myLinkedHashSet = new LinkedHashSet<String>();

Set<String> mySet1 = new HashSet<String>();

myLinkedHashSet.add("Kate");

myLinkedHashSet.add("Mike");

mySet1.add("Mike");

mySet1.add("Kate");

System.out.println("Set: " + mySet1); // Set: [Kate, Mike]

System.out.println("LinkedHashSet: " + myLinkedHashSet); // LinkedHashSet: [Kate, Mike]

Set<String> myTreeSet = new TreeSet<String>();

myTreeSet.add("Chris");

myTreeSet.add("Ann");

System.out.println(myTreeSet); // [Ann, Chris]

}

}

**Collection # 6**

\* 1. Create a HashMap to store two Person Objects into the map: person1 [Name: aaa, Age: 32], person2 [Name: bbb, Age: 45]. Use name as a key String, Person object as the value.

\* 2. Display elements of the HashMap in the square brackets.

\* 3. Display the name person1 if it is in the map.

\* 4. Get a set containing the keys in the map, display all the keys.

\* 5. Get a collection containing the values, display all the values.

\* 6. Create another HashMap to store a few Person Objects.

\* 7. Retrieve the mapping from a HashMap as a Set of Map.entry objects.

\* a. Get a set containing the mappings in this map.

\* b. Iterate through the mappings, display the mappings.

public class ArrayListEx1

{

public static void main(String[] args)

{

Map<String, Person> carMap = new HashMap<String, Person>();

Person person1 = new Person("aaa", 32);

Person person2 = new Person("bbb", 45);

carMap.put(person1.getName(), person1);

carMap.put(person2.getName(), person2);

System.out.println(carMap);

Person foundPerson = carMap.get(person1.getName());

if(foundPerson != null)

System.out.println(foundPerson);

else

System.out.println("The car is not in the set.");

Set<String> keys = carMap.keySet();

Collection<Person> values = carMap.values();

for(String str : keys)

System.out.println(str);

for(Person p : values)

System.out.println(p);

Map<String, Person> carMap1 = new HashMap<String, Person>();

Person person3 = new Person("Name1", 45);

Person person4 = new Person("Name2", 43);

carMap1.put(person3.getName(), person3);

carMap1.put(person4.getName(), person4);

Set<Map.Entry<String, Person>> persons = carMap1.entrySet();

for(Map.Entry<String, Person> entry : persons)

{

System.out.println("Key: " + entry.getKey());

System.out.println("Value: " + entry.getValue());

System.out.println();

}

}

}

**Collection # 7**

/\*\*

\* 1. Create an ArrayList for storing integers.

\* 2. Add Integers 0 - 9 to the list, display the List.

\* 4. Shuffle the elements in the list, display the list.

\* 6. Display the min/max values.

\* 7. Sort the list in the ascending order.

\* 8. Perform the binary search to display number 5.

\*/

import java.util.\*;

class ArrayListEx1

{

public static void main(String[] args)

{

ArrayList<Integer> myList = new ArrayList<Integer>();

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

myList.add(i);

System.out.println(myList); // [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Collections.shuffle(myList); // [8, 0, 2, 5, 6, 3, 1, 7, 9, 4]

System.out.println(myList);

System.out.println("The Minimum value: " + Collections.min(myList)); // The Minimum value: 0

System.out.println("The Maximum value: " + Collections.max(myList)); // The Maximum value: 9

Collections.sort(myList);

System.out.println(myList); // [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

System.out.println(Collections.binarySearch(myList, 5)); // 5

}

}

**ENUM # 1**

/\*\*

\* 1. Declare the enum variable Day that contains 7 days of the week inside the class.

\* 2. Declare a variable weekDay of type Day and assign Monday to it.Display MONDAY [2 ways]. Display: "MONDAY or MONDAY"

\* 3. Display the position of Monday in the enumerated list [2 ways]. Display: "0 or 0"

\* 4. equals method: compare weekDay to Day.MONDAY; display: "The two are the same" if equals

\* 5. compareTo method: compare weekDay to Day.FRIDAY; display "FRIDAY is greater than MONDAY"

\*/

public class EnumEx1

{

enum Day {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY}

public static void main(String[] args)

{

Day weekDay = Day.MONDAY;

System.out.println(weekDay + " or " + Day.MONDAY); // MONDAY or MONDAY

System.out.println(weekDay.ordinal() + " or " + Day.MONDAY.ordinal()); // 0 or 0

if(weekDay.equals(Day.MONDAY))

System.out.println("The two are the same"); // The two are the same

if((Day.FRIDAY).compareTo(weekDay) > 0)

System.out.println("FRIDAY is greater than MONDAY"); // FRIDAY is greater than MONDAY

}

}

**ENUM # 2**

/\*\*

\* 1. Create CarType of an enumerated data type as its own class. [Porsche, Ferrari, Jaguar]

\* 2. Create CarColor of an enumerated data type as its own class. [Red, Blue, Black, Silver]

\* 3. Create SportsCar [Fields: make, color, price; Methods: constructor, getter methods, toString method]

\* 4. Create a car object [porsche, blue, 1000000] and display it. Display dollar amount using DecimalFormatter.

\* 6. Write a method madeIn()- display where was it made with switch statement. [Porsche = Germany, Ferrari = Italy, Jaguar = ENGLAND or unknown CarMaker]

\*/

import java.text.DecimalFormat;

enum CarType{PORSCHE, FERRARI, JAGUAR}

enum CarColor{RED, BLUE, BLACK, SILVER}

class SportsCar

{

private CarType make;

private CarColor color;

private double price;

public SportsCar(CarType make, CarColor color, double price)

{

this.make = make;

this.color = color;

this.price = price;

}

public CarType getMake()

{

return make;

}

public CarColor getColor()

{

return color;

}

public double getPrice()

{

return price;

}

public String toString()

{

DecimalFormat formatter = new DecimalFormat("#,##0.00");

String st = "Car type: " + this.make + "\nCar color: " + this.color + "\nCar price: " + formatter.format(this.price);

return st;

}

}

class ArrayListEx1

{

public static void main(String[] args)

{

SportsCar car = new SportsCar(CarType.PORSCHE, CarColor.BLUE, 100000);

System.out.println(car);

System.out.println(madeIn(car.getMake()));

}

public static String madeIn(CarType type)

{

switch(type)

{

case PORSCHE:

return "Made in Germany";

case FERRARI:

return "Made in Italy";

case JAGUAR:

return "Made in England";

default:

return "Unknown Car Maker";

}

}

}

**ENUM # 3**

/\*\*

\* 1. Declare variable CoffeeSize of type enum as its own class.BIG [big, huge, overwhelming]

\* 2. Declare class Coffee [Fields: size; Methods: constructor, getSize].

\* 3. Create an object and pass size of coffee to to it.

\* 4. Pass 8, 10, 16 ounces to constructor. Write a getOunces() method.

\* 5. Print ounces Using values() method - returns an array of enum values in the order they were created.

\*/

enum CoffeeSize

{

BIG(8), HUGE(10), OVERWHELMING(16);

CoffeeSize(int ounces)

{

this.ounces = ounces;

}

private int ounces;

public int getOunces()

{

return ounces;

}

}

class Coffee

{

private CoffeeSize size;

public Coffee(CoffeeSize size)

{

this.size = size;

}

public CoffeeSize getSize()

{

return size;

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Coffee[] drinks = {new Coffee(CoffeeSize.BIG), new Coffee(CoffeeSize.HUGE), new Coffee(CoffeeSize.OVERWHELMING)};

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

System.out.println("Drink " + (i + 1) + ": " + drinks[i].getSize().getOunces());

for(CoffeeSize cs : CoffeeSize.values())

System.out.println(cs + " " + cs.getOunces());

}

}

**Exception # 1**

/\*\*

\* Create "/home/vladimir/Desktop/MyFile.txt" and insert "24987.62, 28490.9, 29871.44"

\* 1. Create a DecimalFormat Object with the format: "#,##0.00"

\* 3. Read the numbers from the file. Display the total sales, the number of months, and the average amount.

\* 4. Report a FileNotFound Exception and InputMismatchException and report errors using getMessage Method.

\* 5. Add a finally clause that closes a MyFile.txt

\*/

import java.io.FileNotFoundException;

import java.io.File;

import java.util.Scanner;

import java.text.DecimalFormat;

import java.util.InputMismatchException;

public class ArrayListEx1

{

public static void main(String[] args)

{

String st = "/home/vladimir/Desktop/MyFile.txt";

int month = 0;

double totalSales = 0.0;

DecimalFormat formatter = new DecimalFormat("#,##0.00");

Scanner inputFile = null;

try

{

File myFile = new File(st);

inputFile = new Scanner(myFile);

while(inputFile.hasNext())

{

double oneMonth = inputFile.nextDouble();

totalSales += oneMonth;

month++;

}

System.out.println("Number of months: " + formatter.format(month) + " months.");

System.out.println("Total sales: $" + formatter.format(totalSales));

System.out.println("Average sales: $" + formatter.format(totalSales/month));

}

catch(FileNotFoundException e)

{

System.out.println("The file " + st + " does not exist: " + e.getMessage());

}

catch(InputMismatchException e)

{

System.out.println("Non-numeric data found in the file." + e.getMessage());

}

finally

{

inputFile.close();

}

}

}

**Exception # 2**

/\*\*

\* 1. Create a DecimalFormat Object with the format: "#,##0.00"

\* 2. Create openFile() method that opens a file and reports a FileNotFoundException and returns scanner object of the file.

\* 3. Create readFileData() method accepts scanner object and reads in months, sales, return an anonymous double array containing months and total sales.

\* 4. Create a printReport method that displays number of months, total sales, average sales.

\* 5. Report a FileNotFound Exception and InputMismatchException and report errors using getMessage Method.

\*/

import java.io.File;

import java.io.FileNotFoundException;

import java.util.InputMismatchException;

import java.util.Scanner;

import java.text.DecimalFormat;

public class ArrayListEx1

{

public static void main(String[] args)

{

Scanner inputFile = openFile("/home/vladimir/Desktop/MyFile.txt");

double[] arr = readFileData(inputFile); // arr[0] months, arr[1] totalSales

printReport(arr[0], arr[1]);

}

public static Scanner openFile(String st)

{

Scanner inputFile = null;

try

{

File myFile = new File(st);

inputFile = new Scanner(myFile);

}

catch(FileNotFoundException e)

{

System.out.println("The file " + st + " does not exist: " + e.getMessage());

}

return inputFile;

}

public static double[] readFileData(Scanner inputFile)

{

int months = 0;

double totalSales = 0.0;

try

{

while(inputFile.hasNext())

{

totalSales += inputFile.nextDouble();

months++;

}

}

catch(InputMismatchException e)

{

System.out.println("Non-numeric data found in the file." + e.getMessage());

}

finally

{

inputFile.close();

}

return new double[] {months, totalSales};

}

public static void printReport(double months, double totalSales)

{

DecimalFormat formatter = new DecimalFormat("#,##0.00");

System.out.println("# months: " + (int) months + " months \nTotal sales: $" + formatter.format(totalSales) + "\nAverage sales: $" + formatter.format(totalSales/months));

}

}

**Exception # 3**

/\*\*

\* Polymorphic Exception Handling.

\* 1. Convert String to int and catch a NumberFormatException.

\* 2. Catch a General Exception using Polymorphism.

\*/

public class ArrayListEx1

{

public static void main(String[] args)

{

try

{

String st = "abcd";

int num = Integer.parseInt(st);

}

catch(NumberFormatException e)

{

System.out.println("Not a valid integer" + e.getMessage()); // Not a valid integer.

}

catch(Exception e)

{

System.out.println("Polymorphic exception" + e.getMessage()); // Polymorphic error. [Not Recommended!]

}

}

}

**Exception # 4**

/\*\*

\* Read information from the file using a throws clause in the method header.

\*/

import java.io.FileNotFoundException;

import java.io.File;

import java.util.Scanner;

public class ArrayListEx1

{

public static void main(String[] args) throws FileNotFoundException

{

readFile();

}

public static void readFile() throws FileNotFoundException

{

File file = new File("/home/vladimir/Desktop/MyFile.txt");

Scanner inputFile = new Scanner(file);

while(inputFile.hasNext())

System.out.println(inputFile.nextLine());

inputFile.close();

}

}

**Exception # 5**

/\*\*

\* 1. Create a method dateComponent that accepts a string containing a date in the form "month/day/year".

\* 2. StringTokenizer object extracts month/day/year from the string.

\* 3. Write a thow stmt that prevents a null argument to be passed to the method, IllegalArgumentException.

\* 4. Print month/day/year.

\*/

import java.util.StringTokenizer;

public class ArrayListEx1

{

public static void main(String[] args)

{

try

{

dateComponent("4/18/2013");

}

catch(IllegalArgumentException e)

{

System.err.println(e.getMessage());

}

}

public static void dateComponent(String st)

{

if(st == null)

throw new IllegalArgumentException("null reference passed to the method.");

StringTokenizer tokenizer = new StringTokenizer(st, "/");

System.out.println("Month: " + tokenizer.nextToken());

System.out.println("Day: " + tokenizer.nextToken());

System.out.println("Year: " + tokenizer.nextToken());

}

}

**Exception # 6**

/\*\*

\* Create your own Exception if the starting balance on the account is negative.

\*/

class NegativeBalance extends Exception

{

public NegativeBalance()

{

super("Error: Negative Starting Balance.");

}

public NegativeBalance(double amount)

{

super("Error: Negative Starting Balance: $" + amount);

}

}

public class ArrayListEx1

{

public static void main(String[] args) throws NegativeBalance

{

try

{

getBalance(-25.0);

}

catch(NegativeBalance e)

{

System.out.println(e.getMessage());

}

}

public static void getBalance(double amount) throws NegativeBalance

{

if(amount < 0)

throw new NegativeBalance(amount);

}

}

**Flow Control #1**

/\*\*

\* 1. Create an integer array containing 5 elements.

\* 2. Display numbers using an enhanced for loop.

\* 3. Display numbers using a for loop.

\*/

public class FlowControl1

{

public static void main(String[] args)

{

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

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

System.out.print(nums[i] + " ");

for(int num : nums)

System.out.print(num + " ");

}

}

**Flow Control #2**

/\*\*

\* 1. Compare characters 'A' and 'B' using if-else statement

\* 2. Change if-else stmt to a ternary boolean expression.

\*/

public class ArrayListEx1

{

public static void main(String[] args)

{

if('A' < 'B')

System.out.println("char 'A' is less char 'B'"); // char 'A' is less char 'B'

else

System.out.println("char 'B' is less char 'A'");

String st = ('A' < 'B') ? "char 'A' is less char 'B'" : "char 'B' is less char 'A'";

System.out.println(st);

}

}

**Flow Control #3**

/\*\*

\* 1. Use the Switch stmt to input 1, 2, 3, 4, 5 from the user. [4 and 5 choice should say "You entered choice 4 or 5!"]

\*/

import java.util.Scanner;

public class ArrayListEx1

{

public static void main(String[] args)

{

Scanner keyboard = new Scanner(System.in);

System.out.print("Make a choice [1, 2, 3, 4, 5]: ");

int input = Integer.parseInt(keyboard.nextLine());

switch(input)

{

case 1:

System.out.println("You entered choice 1!");

break;

case 2:

System.out.println("You entered choice 2!");

break;

case 3:

System.out.println("You entered choice 3!");

break;

case 4:

case 5:

System.out.println("You entered choice 4 or 5!");

break;

default:

System.out.println("Invalid choice!");

}

}

}

**Flow Control #4**

/\*\*

\* 1. Use the labeled stmt for breaking of outside for-loop.

\* Run outside for-loop from 0-10, inside for-loop from 5-15 at the same time.

\* If outside and inside value equals to each other, display "The values are equals." and break out of outer for loop.

\*/

public class ArrayListEx1

{

public static void main(String[] args) // "inner" is useless, just use "break" instead

{

outer:

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

for(int j = 5; j < 15; j++)

if(i == j)

{

System.out.println("The values are equal");

break outer;

}

}

}

**Garbage Collection # 1**

import java.util.Date;

public class ArrayListEx1

{

public static void main(String[] args)

{

Runtime rt = Runtime.getRuntime();

System.out.println("Total JVM memory: " + rt.totalMemory());

System.out.println("Before memory = " + rt.freeMemory());

Date date = null;

for(int i = 0; i < 10000; i++)

{

date = new Date();

date = null;

}

System.out.println("After Memory = " + rt.freeMemory());

rt.gc();

System.out.println("After GC memory = " + rt.freeMemory());

}

}

**Generics # 1**

/\*\*

\* 1. Create a class Point2D that has (x, y) of generic type T - Number/subclasses. [Fields: x, y. Methods: set/get for x/y, toString]

\* 2. Create an object pt1 of type double with values x = 2.5 and y = 3.5.

\* 3. Write a printAnyTypePoint() method that will prints (Number and its subclasses).

\* 4. Extend Point2D, create 3D point (x, y, z coordinates). [Field z; Method: constructor, to.String set/get for z].

\* 5. Create an object pt2 of type int with values x = 2, y = 3, z = 4. Display it.

\* 6. Write a sum() method to add 2 generic numbers. Test it.

\* 7. Declare an array of generic type.

\*/

class Point2D<T extends Number>

{

private T x;

private T y;

public Point2D(T x, T y)

{

this.x = x;

this.y = y;

}

public void setX(T x)

{

this.x = x;

}

public void setY(T y)

{

this.y = y;

}

public T getX()

{

return this.x;

}

public T getY()

{

return this.y;

}

public String toString()

{

String st = "(" + this.x + ", " + this.y + ")";

return st;

}

}

class Point3D<T extends Number> extends Point2D<T>

{

private T z;

public Point3D(T x, T y, T z)

{

super(x, y);

this.z = z;

}

public void setZ()

{

this.z = z;

}

public T getZ()

{

return z;

}

public String toString()

{

String st = "(" + this.getX() + ", " + this.getY() + ", " + this.z + ")";

return st;

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Point2D<Double> pt1 = new Point2D<Double>(2.5, 3.5);

printAnyTypePoint(pt1);

System.out.println(new Point3D<Integer>(2, 3, 4));

System.out.println("Sum: " + sum(3.5, 4.5));

// E[arr] = (E[]) new Object[10];

}

public static <T extends Number> void printAnyTypePoint(Point2D<T> point)

{

System.out.println(point);

}

public static <T extends Number> double sum(T x, T y)

{

return x.doubleValue() + y.doubleValue();

}

}

**Generics # 2**

public class ArrayListEx1

{

public static void main(String[] args)

{

String[] strArr = {"Chris", "Ann"};

printArr(strArr); // Chris Ann

Integer[] intArr = {1, 2};

printArr(intArr); // 1 2

System.out.println("Sum: " + sumArr(intArr)); // Sum: 3.0

}

public static <E> void printArr(E[] arr)

{

for(E st : arr)

System.out.println(st);

}

public static <E extends Number> double sumArr(E[] arr)

{

double sum = 0;

for(E num : arr)

sum += num.doubleValue();

return sum;

}

}

**Generics # 3**

/\*\*

\* 1. Create a generic class Pair [Fields: first extends number, second String. Methods: constr, get/set, toString].

\* 2. Create a pair object: Integer - 25, String - "Chris". Display the results.

\*/

class Pair<T extends Number, S>

{

private T first;

private S second;

public Pair(T first, S second)

{

this.first = first;

this.second = second;

}

public void setFirst(T first)

{

this.first = first;

}

public void setSecond(S second)

{

this.second = second;

}

public T getFirst()

{

return first;

}

public S getSecond()

{

return second;

}

public String toString()

{

String st = "Number: " + this.first + "\nString: " + this.second;

return st;

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Pair<Integer, String> pair = new Pair<Integer, String>(25, "Chris");

System.out.println(pair); // Number: 25 String: Chris

}

}

**Generics # 4**

/\*\*

\* 1. Create a String array and Integer array to sequentially search an array for a value.

\* 2. Write a searchArr() overloaded methods of E - first search for Strings, second search for Integer.

\* 3. Write a method header search() that accepts type arguments that implement the comparable interface.

\*/

public class ArrayListEx1

{

public static void main(String[] args)

{

String[] strArr = {"One", "Two", "Three"};

System.out.println(searchArr(strArr, "Two")); // Found

Integer[] numArr = {1, 2, 3};

System.out.println(searchArr(numArr, 1)); // Found

}

public static <E> String searchArr(E[] arr, String searchValue)

{

for(E element : arr)

if(element.equals(searchValue))

return "Found";

return "Not found";

}

public static <E extends Number> String searchArr(E[] arr, double searchValue)

{

for(E element : arr)

if(element.doubleValue() == searchValue)

return "Found";

return "Not found";

}

public static <E extends Comparable<E>> void search(){}

}

**Generics # 5**

/\*\*

\* Autobox the value, unbox the object. int to Integer to int

\*/

public class ArrayListEx1

{

public static void main(String[] args)

{

Integer intObject = 7; // Boxing the value

int intPrim = intObject; // Autoboxing the value

System.out.println("Object: " + intObject + "\tPrimitive: " + intPrim); // Object: 7 Primitive: 7

}

}

**Interfaces # 1**

/\*\*

\* 1. Create an interface RetailItem [containing public double getRetailPrice(), getTitle, and constant x(not used)]

\* 2. Create CompactDisc class containg CD Title and a retail price, write get/set methods.

\* 3. Create DvdMovie class containg DVD Title and a retail price, write get/set methods.

\* 4. Create a polymorphic interface array of one CD and DVD, get the title and retail price.

\*/

interface RetailItem

{

public String getTitle();

public double getRetailPrice();

public static final int X = 1;

}

class CompactDisc implements RetailItem

{

private String title;

private double retailPrice;

public CompactDisc(String title, double retailPrice)

{

this.title = title;

this.retailPrice = retailPrice;

}

public String getTitle()

{

return title;

}

public double getRetailPrice()

{

return retailPrice;

}

}

class DvdMovie implements RetailItem

{

private String title;

private double retailPrice;

public DvdMovie(String title, double retailPrice)

{

this.title = title;

this.retailPrice = retailPrice;

}

public String getTitle()

{

return title;

}

public double getRetailPrice()

{

return retailPrice;

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

RetailItem[] items = {new CompactDisc("cdTitle", 10.00), new DvdMovie("dvdTitle", 15.00)};

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

{

System.out.println("Title " + (i + 1) + ": " + items[i].getTitle());

System.out.println("Price " + (i + 1) + ": $" + items[i].getRetailPrice());

}

}

}

**Java IO # 1**

/\*\*

\* 1. Write a method - writeFile1 - to write data to a file "MyTextFile.txt".

\* 2. Write a method - appendFile - to append data to a file.

\* 3. Write a method - readFile - to read data from a file.

\* 4. Write a method - writeFile2 - to write data to a file, make sure the specified file does not exist before opening it.

\* 5. Write a method - analyzeFolder - get the number of files, number of foleders, number of txt files.

\*/

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.io.FileWriter;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class ArrayListEx1

{

public static void main(String[] args)

{

String fileName = "/home/vladimir/Desktop/MyTextFile.txt";

String fileDirectory = "/home/vladimir/Desktop";

writeFile1(fileName);

appendFile(fileName);

readFile(fileName);

writeFile2(fileName);

analyzeFolder(fileDirectory);

}

public static void writeFile1(String fileName)

{

String st = "Hello! My name is not Chris!";

try

{

PrintWriter outputFile = new PrintWriter(fileName);

outputFile.println(st);

outputFile.flush();

outputFile.close();

}

catch(FileNotFoundException e)

{

System.err.println("File does not exist.");

}

}

public static void appendFile(String fileName)

{

String st = "Appended data: Hello! My name is Chris!";

try

{

FileWriter fileWriter = new FileWriter(fileName, true);

PrintWriter outputFile = new PrintWriter(fileWriter);

outputFile.println(st);

outputFile.flush();

outputFile.close();

}

catch(IOException e)

{

System.err.println("File does not exist.");

}

}

public static void readFile(String fileName)

{

try

{

File myFile = new File(fileName);

Scanner inputFile = new Scanner(myFile);

while(inputFile.hasNext())

System.out.println(inputFile.nextLine());

inputFile.close();

}

catch(FileNotFoundException e)

{

System.err.println("File does not exist.");

}

}

public static void writeFile2(String fileName)

{

String st = "Hello! My name is not Chris!";

try

{

File myFile = new File(fileName);

if(myFile.exists())

{

System.out.printf("The file %s already exists\n", fileName);

System.exit(0);

}

PrintWriter outputFile = new PrintWriter(fileName);

outputFile.println(st);

outputFile.flush();

outputFile.close();

}

catch(FileNotFoundException e)

{

System.err.println("File does not exist.");

}

}

public static void analyzeFolder(String fileDirectory)

{

int fileNum = 0;

int textFileNum = 0;

int directoryNum = 0;

File folder = new File(fileDirectory);

File[] listOfFiles = folder.listFiles();

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

{

if(listOfFiles[i].isFile())

fileNum++;

if(listOfFiles[i].isDirectory())

directoryNum++;

if(listOfFiles[i].getName().endsWith(".txt"))

textFileNum++;

}

System.out.printf("There are %d files, %d folders, %d .txt files in the [%s] directory.\n", fileNum, textFileNum, directoryNum, fileDirectory);

}

}

**Java IO # 2**

/\*\*

\* 1. Create a TestScoreReaderFile [Fields: scanner, String] - to read test scores from txt file, split String, calculate avg of each line.

\* 2. The constructor should open a file to read the grades from.

\* 3. readNextLine method reads the next line from the text file.

\* 4. getAverage method calculates the average of the last set of scores from the file.

\* 5. close method Closes the text file

\*/

import java.io.\*;

import java.util.Scanner;

class TestScoreReader

{

private Scanner inputFile;

private String line;

public TestScoreReader(String fileName) throws FileNotFoundException

{

File file = new File(fileName);

inputFile = new Scanner(file);

}

public boolean readNextLine()

{

boolean lineRead = inputFile.hasNext();

if(lineRead)

line = inputFile.nextLine();

return lineRead;

}

public double getAverage()

{

int total = 0;

String[] tokens = line.split(",");

for(String str : tokens)

total += Integer.parseInt(str);

return (double) total / tokens.length;

}

public void close() throws IOException

{

inputFile.close();

}

}

public class ArrayListEx1

{

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

{

int studentNumber = 1;

double avg;

TestScoreReader scoreReader = new TestScoreReader("/home/vladimir/Desktop/NumFile.txt");

while(scoreReader.readNextLine())

{

avg = scoreReader.getAverage();

System.out.println("Average for student " + studentNumber + " is " + avg);

studentNumber++;

}

scoreReader.close();

System.out.println("No more scores");

}

}

**Math Classes**

/\*\*

\* 1. Math class [PI, pow, sqrt].

\* 2. Cast operators for primitive variables [widening vs. narrowing].

\* 3. Post-increment vs. Pre-increment.

\* 4. Difference between 6/4 and 6.0/4

\* 5 .Creating a constant inside/outside a method.

\* 7. Create a Random object [A-F].

\* 8. Create a DecimalFormat object and display [000.17 001.67 016.67 166.67]

\* 9. Create a DecimalFormat object and display [123.45 1,234.56 1,234,567.89]

\* 10. Create a DecimalFormat object and convert a double value to percent.

\* 11. printf method for formatting. Display 1: 9 1.20 Hello " 1.60" Display 2: Your pay is 1,253,874.12

\* 12. Validate number input entered by the user using Scanner methods.

\*/

import java.util.Scanner;

import java.util.Random;

import java.text.DecimalFormat;

public class ArrayListEx1

{

public static final double PI = 3.14;

public static void main(String[] args)

{

System.out.println("Pi: " + Math.PI); // Pi: 3.141592653589793

System.out.println("2 to the power 3: " + Math.pow(2, 3)); // 2 to the power 3: 8.0

System.out.println("Square root of 16: " + Math.sqrt(16)); // Square root of 16: 4.0

int num1 = 5; // Widening

double num2 = num1;

System.out.println(num1 + num2); // 10

double num3 = 5; // Narrowing

int num4 = (int) num3;

System.out.println(num3 + num4); // 10

int num5 = 5;

System.out.println((num5++) + " " + (++num5)); // 5 7

System.out.println(6/4 + " vs " + 6.0/4); // 1 vs 1.5

final double CONSTANT = 7.28;

Random random = new Random();

char[] chars = {'A', 'B', 'C', 'D', 'E', 'F'};

System.out.println(chars[random.nextInt(6)]); // A - F

DecimalFormat formatter1 = new DecimalFormat("000.00");

double[] numArr1 = {0.16667, 1.6667, 16.667, 166.67};

for(int i = 0; i < numArr1.length; i++)

System.out.println(formatter1.format(numArr1[i]));

DecimalFormat formatter2 = new DecimalFormat("#,##0.00");

double[] numArr2 = {123.45, 1234.56, 1234567.89};

for(int i = 0; i < numArr2.length; i++)

System.out.println(formatter2.format(numArr2[i]));

double num6 = 0.12, num7 = 0.05;

DecimalFormat formatter = new DecimalFormat("#0%");

System.out.println(formatter.format(num6) + "\t" + formatter.format(num7)); // 12% 5%

int num8 = 9;

System.out.printf("%d %.2f %s \"%6.2f\"\n", num8, 1.2, "Hello", 1.6); // 9 1.20 Hello " 1.60"

System.out.printf("Your pay is %,.2f\n", 1253874.12); // Your pay is 1,253,874.12

Scanner keyboard = new Scanner(System.in);

System.out.print("Enter a valid number: ");

while(!keyboard.hasNextDouble())

{

System.out.print("Wrong input! Enter a valid number: ");

keyboard.nextLine();

}

System.out.println("You entered: " + Double.parseDouble(keyboard.nextLine()));

}

}

**Bank Account**

/\*\*

\* 1. Create class BankAccount [Fields: balance. Methods: constr. w/balance, constr. parsing string, deposit(), withdrawal, set/getBalance(), toString()].

\* 2. Create your own exception class if the object BankAccount with a Negative Starting Balance [throw exception in constr and setBalance()].

\* 3. Write a static method createBankAccount() for creating BankAccount objects. Write a checkBalance() method to check for the negative balance.

\* 4. Create an array of 2 bank accounts and display its deposit using print format.

\* 5. Write totalBankBalance() method using vararg type to get a total balance of the bank of 2 bank accounts.

\* 6. Create an ArrayList of BankAccounts with cap. 100, add 2 bank accounts display balance for each account.

\* 7. Write a method GarbageCollector() Use a JVM Garbage Collection for removal of unreferenced objects from memory.

\*/

import java.util.ArrayList;

import java.text.DecimalFormat;

class NegativeBalance extends Exception

{

public NegativeBalance()

{

super("Error: Negative Starting Balance.");

}

public NegativeBalance(double amount)

{

super("Error: Negative Starting Balance: $" + amount);

}

}

class BankAccount

{

private double balance;

public BankAccount()

{

this.balance = 0.0;

}

public BankAccount(double startBalance)

{

this.balance = startBalance;

}

public BankAccount(String str)

{

this.balance = Double.parseDouble(str);

}

public void deposit(double amount)

{

balance += amount;

}

public void withdraw(double amount)

{

balance -= amount;

}

public void setBalance(double balance)

{

this.balance = balance;

}

public void setBalance(String str)

{

this.balance = Double.parseDouble(str);

}

public double getBalance()

{

return this.balance;

}

public String toString()

{

String st = "Balance: " + this.balance;

return st;

}

public static double totalBankBalance(BankAccount...accounts)

{

double totalBalance = 0;

for(BankAccount acc : accounts)

totalBalance += acc.getBalance();

return totalBalance;

}

}

public class ArrayListEx1

{

public static void main(String[] args) throws NegativeBalance

{

DecimalFormat formatter = new DecimalFormat("#,###.00");

BankAccount[] accounts = {createBankAccount(100), createBankAccount(200)};

System.out.println("Here are the balances for each account from the array: ");

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

System.out.printf("The balance for account # %d is $%6.2f\n", (i + 1), accounts[i].getBalance());

System.out.printf("The total balance of the bank accounts is $%6.2f\n", BankAccount.totalBankBalance(createBankAccount(100), createBankAccount(1300)));

ArrayList<BankAccount> accts = new ArrayList<BankAccount>(100);

accts.add(createBankAccount(500));

accts.add(createBankAccount(600));

for(BankAccount acc : accts)

System.out.println("The balance is $" + formatter.format(acc.getBalance()));

garbageCollector(createBankAccount(10));

}

public static void garbageCollector(BankAccount account)

{

BankAccount tempAccount = account;

account = null;

tempAccount = null;

}

public static void checkBalance(double amount) throws NegativeBalance

{

if(amount < 0)

throw new NegativeBalance(amount);

}

public static BankAccount createBankAccount(double amount) throws NegativeBalance

{

try

{

checkBalance(amount);

}

catch(NegativeBalance e)

{

System.err.println(e.getMessage());

System.exit(0);

}

return new BankAccount(amount);

}

}

**Graded Activity**

/\*\*

\* FinalExam -> GradedActivity

\* GradedActivity: a) Create class GradedActivity [Field: score: get/setScore].

\* GradedActivity: b) Write a getGrade() method - 'A' if above 90, 'B' above 80, 'C' above 70

\* FinalExam [inherits GradedActivity]: [Fields: numQuestions, numMissed; Method: constr, getPointsEach, getNumMissed(), getNumQuestions() get the numberic score and pass it to the super GradedActivity setScore method from the constructor]

\* FinalExam: display number of questions, number of questions missed, points of each question, score and grade for the final exam.

\* Create an array of GradedActivity references: new GradedActivity(); setScore to 95, new FinalExam [50 questions and 7 missed]

\* Display score and grade for each test.

\* Create a new GradedActivity object and determine if it references a GradedActivity object using instanceof operator.

\*/

class GradedActivity

{

private double score;

public void setScore(double score)

{

this.score = score;

}

public double getScore()

{

return this.score;

}

public String getGrade()

{

if(this.score >= 90)

return "A";

else if(this.score >= 80)

return "B";

else if(this.score >= 70)

return "C";

else if(this.score >= 70)

return "D";

else

return "F";

}

}

class FinalExam extends GradedActivity

{

private int numQuestions;

private int numMissed;

private int pointsEach;

public FinalExam(int numQuestions, int numMissed)

{

this.numQuestions = numQuestions;

this.numMissed = numMissed;

this.pointsEach = 100 / numQuestions;

super.setScore((numQuestions - numMissed) \* pointsEach);

}

public double getPointsEach()

{

return pointsEach;

}

public double getNumMissed()

{

return numMissed;

}

public double getNumQuestions()

{

return numQuestions;

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

FinalExam exam = new FinalExam(100, 25);

System.out.println("Number of questions: " + exam.getNumQuestions());

System.out.println("Number of questions missed: " + exam.getNumMissed());

System.out.println("Points for each question: " + exam.getPointsEach());

System.out.println("Total score: " + exam.getScore());

System.out.println("Final Grade: " + exam.getGrade());

GradedActivity[] activities = {new GradedActivity(), new FinalExam(50, 7)};

activities[0].setScore(95);

for(GradedActivity activity : activities)

{

System.out.println("Score: " + activity.getScore() + "\tGrade: " + activity.getGrade());

if(activity instanceof GradedActivity)

System.out.println("Instance of GradedActivity");

}

}

}

**Rectangle**

/\*\*

\* 1. Create a Class Rectangle [Fields: length, width; Methods: constructor, set/get length and width, getArea, toString]

\* 2. Write a method createRectangle() for creating objects.

\* 3. Write a method displayRectangle() to display a method to display rectangle.

\* 7. Write a method to change dimensions of the rectangle changeRectangle().

\* 8. Create a Cube class that extends a Rectangle class [Fields: length, width, height; Methods: set/getHeight, getSurfaceArea (area \* 6), getVolume(area \* h), toString]

\*/

class Rectangle

{

private double length;

private double width;

public Rectangle(double length, double width)

{

this.length = length;

this.width = width;

}

public void setLength(double length)

{

this.length = length;

}

public void setWidth(double width)

{

this.width = width;

}

public double getLength()

{

return this.length;

}

public double getWidth()

{

return this.width;

}

public double getArea()

{

return this.length \* this.width;

}

public String toString()

{

String st = "Length: " + this.length + "\tWidth: " + this.width + "\tArea: " + this.getArea();

return st;

}

}

class Cube extends Rectangle

{

private double length, width, height;

public Cube(double length, double width, double height)

{

super(length, width);

this.height = height;

}

public void setHeigth(double height)

{

this.height = height;

}

public double getHeight()

{

return this.height;

}

public double getSurfaceArea()

{

return getArea() \* 6;

}

public double getVolume()

{

return getArea() \* height;

}

public String toString()

{

String st = super.toString() + "\tHeight: " + this.getHeight() + "\tSurface Area: " + this.getSurfaceArea() + "\tVolume: " + this.getVolume();

return st;

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Rectangle rectangle1 = createRectangle(5.0, 5.0);

displayRectangle(rectangle1); // Length: 5.0 Width: 5.0 Area: 25.0

changeRectangle(rectangle1, 6.0, 6.0);

displayRectangle(rectangle1); // Length: 6.0 Width: 6.0 Area: 36.0

Cube cube = new Cube(1, 2, 3);

System.out.println(cube);

}

public static Rectangle createRectangle(double length, double width)

{

return new Rectangle(length, width);

}

public static void displayRectangle(Rectangle rectangle)

{

System.out.println(rectangle);

}

public static void changeRectangle(Rectangle rectangle, double length, double width)

{

rectangle.setLength(length);

rectangle.setWidth(width);

}

}

**Polymorphism**

/\*\*

\* Create a super class Animal. Create Cat and Dog subclasses that extend from Animal Class.

\* Add methods makeNoice() for each class: Animal - "No Sound", Cat - "Meow", "Dog" - "ruff"

\* Create an array of Animals consisting of Cat and Dog objects and call a makeNoice() for each object.

\*/

class Animal

{

public void makeNoice()

{

System.out.println("No Sound!");

}

}

class Cat extends Animal

{

public void makeNoice()

{

System.out.println("Meow!");

}

}

class Dog extends Animal

{

public void makeNoice()

{

System.out.println("Ruff!");

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Animal[] animals = {new Dog(), new Cat()};

for(Animal a : animals)

a.makeNoice();

}

}

**Regex**

/\*\*

\* Find "ab" in "abaaba" using regular expressions.

\* Find a hexadecimal number.

\* Find 7 digits in a row or 3 digits followed by a dash or a space followed by 4 digits.

\* Find 2-20 character word.

\* Find zip-code 5 digits long.

\* Find two states that start with C and A.

\* Find min of four star symbols.

\* Find an email.

\*/

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class ArrayListEx1

{

public static void main(String[] args)

{

regexEngine("ab", "abaaaaba");

regexEngine("0[xX][0-9a-fA-F]", "0x09aF");

regexEngine("\\d\\d\\d([\\s])?\\d\\d\\d\\d", "123 45678");

regexEngine("[a-zA-Z]{2,20}"," ascdef");

regexEngine("\\s[\\d]{5}", " 12345");

regexEngine("\\s[CA][AKLRZ]\\s", " AZ ");

regexEngine("[A-Za-z0-9\_%.-]+@[A-Za-z0-9\_%.-]+\\.[A-Za-z]{2,4}", "mail@mail.com");

regexEngine("\\\*{4,}", "\*\*\*\*\*");

}

public static void regexEngine(String expression, String text)

{

Pattern pattern = Pattern.compile(expression);

Matcher matcher = pattern.matcher(text);

System.out.println("Pattern is " + matcher.pattern());

while(matcher.find())

System.out.println(matcher.start() + " " + matcher.group());

}

}

**Serializatiom**

/\*\*

\* Object serialization.

\*/

import java.io.Serializable;

import java.io.IOException;

import java.io.FileOutputStream;

import java.io.ObjectOutputStream;

class Person implements Serializable

{

private int age;

private String name;

public Person(String name, int age)

{

this.age = age;

this.name = name;

}

public void setAge(int age)

{

this.age = age;

}

public void setName(String name)

{

this.name = name;

}

public int getAge()

{

return this.age;

}

public String getName()

{

return name;

}

}

public class SerializeObject

{

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

{

Person[] persons = {new Person("Chris", 25), new Person("Ann", 24), new Person("Mary", 27)};

FileOutputStream outStream = new FileOutputStream("/home/vladimir/Desktop/Objects.dat");

ObjectOutputStream objectOutputFile = new ObjectOutputStream(outStream);

for(Person p : persons)

objectOutputFile.writeObject(p);

objectOutputFile.close();

System.out.println("The Serialized Objects were written to the Objects.dat file.");

}

}

**Deserialization**

/\*\*

\* Object deserialization.

\*/

import java.io.Serializable;

import java.io.IOException;

import java.io.FileInputStream;

import java.io.ObjectInputStream;

public class DeserializeObject

{

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

{

FileInputStream inStream = new FileInputStream("/home/vladimir/Desktop/Objects.dat");

ObjectInputStream objectInputFile = new ObjectInputStream(inStream);

Person[] persons = new Person[3];

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

persons[i] = (Person) objectInputFile.readObject();

objectInputFile.close();

for(Person p : persons)

System.out.println(p);

}

}

**Strings**

/\*\*

\* 1. Concatenate two strings: "Hello " and "Chris" and display the length.

\* 2. Concatenate "Greeting: " and "Hello Chris" using concat string method.

\* 4. Display the first character of the string, display the index of the character 'e'.

\* 5. Display String in a lower and upper case.

\* 6. Memory address vs. meaningful equality.

\* 7. Make one string equal to another String, make a change to the second string. What is the effect on the original string?

\* 8. Convert String input to an integer.

\* 9. Split a string using a split method and display an array of Strings. [Hello, Chris, Kelly]

\* 10. Show an example substring method. String "Chris Kelly", return "Kelly"; String "Hello my Hello" return "my"

\* 11. Replace all 'v' chars with 'b': "vova" -> "boba"

\* 13. Trim the spaces in front/before string: \* Hi \* -> \*Hi\*

\* 14. StringBuilder append "Hello" and "Chris". Display the StringBuilder with toString() method.

\* 15. StringBuilder insert method: 9195551212 to (919)555-1212

\* 16. StringBuilder t method: "We moved from Chicago" to "We moved from New York"

\* 17. StringBuilder delete method: "I ate 100 strawberries" to "I ate strawberries"

\* 18. StringTokenizer string method: Tokenize "One Two Three Four"

\* 19. Use String split method to split 2 delimeters. Split "name1@name2.com" into "name1 name2 com"

\*/

import java.util.StringTokenizer;

public class ArrayListEx1

{

public static void main(String[] args)

{

String st1 = "Hello " + "Chris";

String st2 = "Hi";

System.out.println(st1 + " Length: " + st1.length()); // Hello Chris Length: 11

System.out.println(("Greeting: ").concat(st1)); // Greeting: Hello Chris

System.out.println("First character: " + st1.charAt(0) + " The index of 'e': " + st1.indexOf('e')); // First character: H The index of 'e': 1

System.out.println("Lower case: " + st1.toLowerCase() + " Upper case: " + st1.toUpperCase()); // Lower case: hello chris Upper case: HELLO CHRIS

System.out.println(Integer.parseInt("123")); // 123

System.out.println("vova" + " -> " + ("vova").replace('v', 'b')); // boba

System.out.println("\*" + " Hi " + "\*" + " -> " + "\*" + (" Hi ").trim() + "\*"); // \* Hi \* -> \*Hi\*

System.out.println(("Chris Kelly").substring(6) + " " + ("Hello my Hello").substring(6, 8)); // Kelly my

if(st1 != st2)

System.out.println("Two strings have different memory address"); // Two strings have different memory addresses

if(!st1.equals(st2))

System.out.println("Two strings are not meaningfully equal"); // Two strings are not meaningfully equal

String st3 = "Hello";

String st4 = st3;

st4 = "Hi";

System.out.println(st3); // Hello

String st5 = "Hello, Chris, Kelly";

String[] strArr = st5.split(", ");

for(String st : strArr) // Hello Chris Kelly

System.out.println(st);

String st6 = "name1@name2.com";

String[] stArr = st6.split("[@.]");

for(String st : stArr) // name1 name2 com

System.out.println(st);

StringTokenizer tokenizer = new StringTokenizer("One Two Three Four", " ");

while(tokenizer.hasMoreTokens())

System.out.println(tokenizer.nextToken()); // One Two Three Four

StringBuilder builder1 = new StringBuilder();

builder1.append("Hello ");

builder1.append("chris");

System.out.println(builder1.toString()); // Hello chris

StringBuilder builder2 = new StringBuilder();

builder2.append("9195551212");

builder2.insert(0, "(");

builder2.insert(4 , ")");

builder2.insert(8, "-");

System.out.println(builder2.toString()); // (919)555-1212

StringBuilder builder3 = new StringBuilder("We moved from Chicago");

builder3.replace(14, 21, "New York");

System.out.println(builder3.toString()); // We moved from New York

StringBuilder builder4 = new StringBuilder("I ate 100 strawberries");

builder4.delete(6, 10);

System.out.println(builder4.toString()); // I ate strawberries

}

}

**Thread # 1**

/\*\*

\* Create class Dog that implements Runnable interface. [Field: name, random time; Method: run]

\* run method: display the name of the dog, the name of the thread, unique thread id, priority of the thread and put the thread to sleep for the random time.

\* Create an array of two thread objects and set the name, priority of each. Run it, display the results.

\*/

import java.util.Random;

class Dog implements Runnable

{

private int time;

private String name;

Random random = new Random();

public Dog(String name)

{

this.name = name;

this.time = random.nextInt(1000);

}

public void run()

{

System.out.println("The name of the thread: " + Thread.currentThread().getName());

System.out.println("The ID of the thread is " + Thread.currentThread().getId());

System.out.println("The priority of the thread is " + Thread.currentThread().getPriority());

System.out.println("The name of dog is " + this.name);

try

{

Thread.sleep(time);

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Thread[] threads = {new Thread(new Dog("Dog1")), new Thread(new Dog("Dog2"))};

threads[0].setName("Thread1");

threads[1].setName("Thread2");

threads[0].setPriority(1);

threads[1].setPriority(2);

for(Thread thread : threads)

thread.start();

}

}

**Thread # 2**

/\*\*

\* Create a class Runner [extends Thread class] and run it. Display the name of the thread and put it to sleep for 100 ms.

\*/

class Runner extends Thread

{

public void run()

{

System.out.println("Currently running thread: " + Thread.currentThread().getName());

try

{

Thread.sleep(100);

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Runner[] runners = {new Runner(), new Runner()};

for(Runner r : runners)

r.start();

}

}

**Thread # 3**

/\*\*

\* Code is running in its own thread [run the interface runnable].

\* Display the name of the thread and put it to sleep for 1s.

\*/

public class ArrayListEx1

{

public static void main(String[] args)

{

Thread thread = new Thread(new Runnable(){

public void run()

{

System.out.println("The name of the thread: " + Thread.currentThread().getName());

try

{

Thread.sleep(1000);

}

catch (InterruptedException e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

}

});

thread.start();

}

}

**Thread # 4**

/\*\*

\* Create a class Processor that extends thread [Fields: boolean running set to true], run method run() printing "Hello" infinitely until running is false, put the thread to sleep for 100ms.

\* Write a method shutdown() to shutdown the while loop that sets running to false.

\*/

import java.util.\*;

class Processor extends Thread

{

private volatile boolean running = true;

public void run()

{

while(running)

{

System.out.println("Hello");

try

{

Thread.sleep(1000);

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

public void shutDown()

{

running = false;

}

}

public class ArrayListEx1

{

public static void main(String[] args)

{

Processor processor1 = new Processor();

processor1.start();

System.out.println("Press return to stop ...");

Scanner keyboard = new Scanner(System.in);

keyboard.nextLine();

processor1.shutDown();

}

}

**Thread # 5**

/\*\*

\* Create a class Account [Field: balance of 50; Method: getBalance, synchronized withdraw]

\* write a synchronized withdraw() method that substracts $10 from the balance.

\* Write a synchronized method makeWithdrawal which displays the name of the thread in the bank. Check if getBalance > 10 [puts the thread to sleep, call withdraw method, display message "The account has been withdrawn"] otherwise display the name - and say not enough money is in the bank.

\* run() method - run makeWithrdawal five times with for-loop.

\* Fred and Lucy have both access to the same account object, setName of the threads to Lucy and Fred respectively.

\*/

class Account implements Runnable

{

private double balance = 50;

public double getBalance()

{

return balance;

}

public synchronized void withdraw()

{

balance -= 10;

}

private synchronized void makeWithdrawal()

{

if(getBalance() > 10)

{

System.out.println(Thread.currentThread().getName() + " is in the bank");

try

{

Thread.sleep(500);

}

catch (InterruptedException e)

{

e.printStackTrace();

}

withdraw();

System.out.println("The account has withdrawn");

}

else

{

System.err.println(Thread.currentThread().getName() + " not enough money!");

}

}

public void run()

{

for(int i = 0; i < 5; i++)

{

makeWithdrawal();

}

}

}

public class ArrayListEx1

{

public static Account account = new Account();

public static void main(String[] args)

{

Thread lucy = new Thread(account);

Thread fred = new Thread(account);

lucy.setName("Lucy");

fred.setName("Fred");

lucy.start();

fred.start();

}

}