Math.sqrt(Math.sqrt(x)); Math.hypot(a, b); Math.max(population1, Math.max(population2, Math.max(population3, population4)))

Write a static method named merge that receives these two arrays as parameters and returns a reference to a new, sorted array of ints that is the result of merging the contents of the two arrays , arr1 and arr2:

public static int[] merge(int[] arr1, int[] arr2)

{ int merged[] = new int[arr1.length + arr2.length];

int index\_arr1 = 0;

int index\_arr2 = 0;

int index\_merged = 0;

while(index\_arr1 < arr1.length || index\_arr2 < arr2.length)

{ if(index\_arr2 >= arr2.length || (index\_arr1 < arr1.length && arr1[index\_arr1] < arr2[index\_arr2]))

{ merged[index\_merged++] = arr1[index\_arr1++];

}

else

{ merged[index\_merged++] = arr2[index\_arr2++];

}

}

return merged;

}

An array of ints , arr, has been declared and initialized . Write the statements needed to reverse the elements in the array . So, if the elements were originally 5, 13, 4, 97 then after your code executes they would be 97, 4, 13, 5:

for (int i = 0; i < arr.length/2; i++)

{ int temp = arr[i];

arr[i] = arr[arr.length - i - 1];

arr[arr.length - i - 1] = temp;

}

Assume you are given an int variable named sum and a 2-dimensional array of ints that has been created and assigned to a2d. Write some statements that compute the sum of all the elements in the entire 2-dimensional array and assign the value to sum:

sum = 0;

for (int i = 0; i < a2d.length; i++)

{ for (int j = 0; j < a2d[i].length; j++)

{ sum += a2d[i][j];

}

}

Assume you are given an int variable named nPositive and a 2-dimensional array of ints that has been created and assigned to a2d. Write some statements that compute the number of all the elements in the entire 2-dimensional array that are greater than zero and assign the value to nPositive.

nPositive = 0;

for (int i = 0; i < a2d.length; i++)

{ for (int j = 0; j < a2d[i].length; j++)

{ if (a2d[i][j] > 0)

{ nPositive++;

}

}

}

Given an existing class , BankAccount, containing:

a constructor accepting a String corresponding to the name of the account holder.

a method , getBalance, that returns a double corresponding to the account balance.

a method withdraw that accepts a double , and deducts the amount from the account balance.

Write a class definition for a subclass, CheckingAccount, that contains:

a boolean instance variable , overdraft. (Having overdraft for a checking account allows one to write checks larger than the current balance).

a constructor that accepts a String and a boolean . The String parameter is used in the invocation of the superclass (BankAccount) constructor , while the boolean is used to initialize the overdraft instance variable .

a method , hasOverdraft, that returns a boolean . hasOverdraft returns true if the account supports overdraft.

a method , clearCheck, that accepts a double and returns a boolean . clearCheck will determine if the amount (of the check) can be cashed-- this will be the case if the amount is less than the balance in the account, or if the account allows overdraft. If the check can be cashed, clearCheck returns true , and also calls the withdraw method to update the account balance; otherwise, clearCheck returns false:

public class CheckingAccount extends BankAccount {

private boolean overdraft;

public CheckingAccount(String accountHolderName, boolean initialOverdraft)

{ super(accountHolderName);

overdraft = initialOverdraft;

}

public boolean hasOverdraft()

{ return overdraft;

}

public boolean clearCheck(double requestedCheckAmount)

{ if (requestedCheckAmount < getBalance() || overdraft)

{ withdraw(requestedCheckAmount);

return true;

}

else

{ return false;

}

}

}

Selection Sort Algorithm

Selection sort conceptually divides the array into two parts: the sorted part at the front and the unsorted part at the end.

• When we start, the unsorted part consists of the entire array.

1. Search the unsorted part of the array and find the smallest

element.

2. Swap this value with the first element in the unsorted part of the array.

3. Decrease the size of the unsorted part of the array by 1.

4. Repeat steps 1 – 3 until only one element remains in the unsorted part.

Bubble Sort

Repeat the following until the array is sorted:

for (index = 0; index < a.length - 1; index++)

{

if( a[index] > a[index + 1])

Interchange the values of a[index] and

a[index + 1]:

(double temp;

temp = a[i];

a[i] = a[j];

a[j] = temp;)

}

Static Methods

• Sometimes a method may have no relation to a specific object.

• Static method are declared in a class

– Can be invoked without using an object – Instead use the class name.

• A static method cannot refer to an instance variable of the class, and it cannot invoke a nonstatic method of the class without creating an object of the class type.

Static Variables

• Static variables also called class variables

– Contrast with instance variables

• Do not confuse class variables with variables of a class type!

• Static variables are shared by all objects of a class

– Only one instance of the variable exists. – It can be accessed by all objects of the class.

• A computer has two forms of memory

• Secondary memory is used to hold files for "permanent" storage (non-volatile)

• Main memory is used by a computer when it is running a program (volatile)

– Values stored in a program's variables are kept in main memory

Variables in Memory

• When the variable is a primitive type, the value of the variable is stored in the memory location assigned to the variable

• When the variable is a class type, only the memory address (or reference) where its object is located is stored in the memory location assigned to the variable

• A method cannot change the value of a variable of a primitive type that is an argument to the method

• In contrast, a method can change the values of the instance variables of a class type that is an argument to the method.

• A copy constructor is a constructor with a single argument of the same type as the class

• The copy constructor should create an object that is a separate, independent object, but with the

instance variables set so that it is an exact copy of the argument object

• A class that contains no methods (other than constructors) that change any of the data in an object of the class is called an immutable class

– Objects of such a class are called immutable objects

– It is perfectly safe to return a reference to an immutable object because the object cannot be changed in any way

– The String class is an immutable class

Mutable and Immutable Classes

• A class that contains no methods (other than constructors) that change any of the data in an

object of the class is called an immutable class

– Objects of such a class are called immutable objects

– It is perfectly safe to return a reference to an immutable object because the object cannot be changed in any way

– The String class is an immutable class

Mutable and Immutable Classes

• A class that contains public mutator methods or other public methods that can change the data in its

objects is called a mutable class, and its objects are

called mutable objects

– Never write a method that returns a mutable object

– Instead, use a copy constructor to return a reference to a completely independent copy of the mutable object

• A deep copy of an object is a copy that, with one exception, has no references in common with the original

– Exception: References to immutable objects are allowed to be shared

• Any copy that is not a deep copy is called a shallow copy

– This type of copy can cause dangerous privacy leaks in a program

Overriding (between base and derived class): Same name; same # and type of parameters

Overloading (within same class/between base and derived class): Same name; different # and/or type of parameters

• An entire class may be declared final (cannot override)

– Thus, cannot be used as a base class to derive any other class

• Java has a class that is the ultimate ancestor of every class

– The class Object

public boolean equals(Object otherObject)//overriding

{ if (otherObject == null)

{ return false;

}

else if (getClass() != otherObject.getClass())

{ return false;

}

else

{ Student otherStudent = (Student)otherObject;

return (sameName(otherStudent)) && studentNumber == otherStudent.studentNumber;

}

}

• Polymorphism is the ability to associate many meanings to one method name

– It does this through a special mechanism known as dynamic binding or late binding

Dynamic Binding

• When an overridden method is invoked:

§ Java first looks for the method in the class that was used to create the object (the class that was used in the call to new).

§ If the method does not exist in that class, it looks up the inheritance hierarchy unAl it finds the method.

§ The method is not determined by the type of the variable.

• Variable of any ancestor class can reference object of descendant class.

§ Object always remember their original type!

Consider an array of Person

Person[] people = new Person[4];

Since Student and Undergraduate are types of Person, we can assign them to Person variables

people[0] = new Student("DeBanque, Robin", 8812);

people[1] = new Undergraduate("Cotty, Manny", 8812, 2);

Upcasting is when an object of a derived class is assigned to a variable of a base class (or any ancestor class)

• Downcas3ng is when a type cast is performed from a base class to a derived class (or from any ancestor class to any descendent class)

• There are Ames, however, when downcasting is necessary, e.g., inside the equals method for a class

• The instanceof operator checks if an object is of the type given as its second argument: “Object instanceof ClassName”

• Abstract classes are like regular classes except that you cannot create an object (instance) of an abstract class using the new operator.

• Abstract methods are methods with no implementation, only a header.

• Not all methods of an abstract class need to be abstract methods.