Classes and Object DB

**What is an object?**

is an instance of a class

it is created wiht the “new”

object class is parent

**What is a class?**

A class is a blueprint used to create objects.

It has a “class” declaration

An access specifier public

Name

{ }

It contains variables and methods.

It hides the data and methods (data encapsulation)

A class with a main method begins the running of a java application

A class can be reused to create other objects

One class has a public static main(String[] args) method that begins the program

It is static so it belongs to the class and not the instances (objects) of the class

**What is the relationship between the two concepts?**

**Objects**

Java programs are made up of classes. There has to be at least one class in a java program.  There can be more than one class in a program and the classes can be grouped into packages.  One of  the java classes in the program has a main method that begins the program. The main method is static so it belongs to the class and not to the instances (objects) of the class. There can only be one public class in a file. Java classes are the blueprint  used to create objects

There is an Object class that is the top class. The Object class does not have a super class. Every class is a direct or indirect descendant of the Object class and inherits the instance methods of Object. All other classes besides the object class has one superclass.

An Object is also an  instance of a class. . The class can be reused to create more than one instance (object) of the class. Objects aer created with the "new" operator which allocates memory for an object.

**Provide an example.**

Java programs are made up of classes that are used to create classes. There has to be at least one class in a java program.  There can be more than one class in a program and the classes can be grouped into packages.  One of  the java classes in the program has a main method that begins the program. The main method is static so it belongs to the class and not to the instances (objects) of the class. There can only be one public class in a file. Java classes are the blueprint used to create objects

There is also a class called Class. Class extends the Object class.

The Object class is the top class. The Object class does not have a superclass. Every class is a direct or indirect descendant of the Object class and inherits the instance methods of Object. Each class besides the Object class has one superclass.

An object is also an  instance of a class. The class can be reused to create more than one instance (object) of the class. Objects are created with the "new" operator which allocates memory for an object.

Objects have the characteristics of "state" and "behavior" that is determined by the class.  The Non-static variables in the class become instance variables in each object that belongs to that class. The data is inside the individual object, not the class. Each object has its own set of data. It has its own "state".

An example is:

public class Lunch

{

    public static void main(String[] args){

        Lunch fridayLunch = new Lunch();

    }

**/\*\***

**\***

**\* @author Bernice**

**\*/**

**public class HugeInteger**

**{**

**@SuppressWarnings("FieldMayBeFinal")**

**private int[] intArray;**

**private int numDigits; // stores the number of digits in intArray**

**// 1 arg constructor**

**public HugeInteger(String s)**

**{**

**intArray = new int[40];**

**numDigits = 0;**

**// call parse(s)**

**this.parse(s);**

**}**

**//no-arg constructor**

**public HugeInteger()**

**{**

**intArray = new int[40];**

**numDigits = 0;**

**}**

**//method to parse string**

**public void parse(String s)**

**{**

**// Add each digit to the arrays**

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

**{**

**intArray[39-i] = (int) s.charAt(i);**

**}**

**// update numDigits**

**numDigits++;**

**}**

**public static HugeInteger add(HugeInteger hugeInt1, HugeInteger hugeInt2)**

**{**

**// Create hugeInt3**

**HugeInteger hugeInt3 = new HugeInteger();**

**// Loop**

**for (int i = 39; i >= 0; i--)**

**{**

**// Add digits from hugeInt1 and hugeInt2,**

**// Store in corresponding hugeInt3**

**if ((hugeInt1.intArray[i] + hugeInt2.intArray[i]) < 10)**

**{**

**hugeInt3.intArray[i]= hugeInt1.intArray[i] + hugeInt2.intArray[i];**

**}**

**else**

**{**

**hugeInt3.intArray[i]= (hugeInt1.intArray[i] + hugeInt2.intArray[i]-10);**

**hugeInt1.intArray[i-1]++;**

**}**

**// End**

**}**

**//**

**// return hugeInt3**

**return hugeInt3;**

**}**

**//this is static**

**public static HugeInteger subtract(HugeInteger hugeInt1, HugeInteger hugeInt2)**

**{**

**// Create hugeInt3**

**HugeInteger hugeInt3 = new HugeInteger();**

**// Loop**

**// Subtract hugeInt2 digit from hugeInt1,**

**// Store in corresponding hugeInt3**

**// End**

**//**

**// return hugeInt3**

**return hugeInt3;**

**}**

**public static boolean isEqualTo(HugeInteger hugeInt1, HugeInteger hugeInt2)**

**{**

**// return true if the value represented by**

**// elements of hugeInt1.intArray is equal to**

**// value represented by elements of hughInt2.intArray**

**return true;**

**}**

**public static boolean isNotEqualTo(HugeInteger hugeInt1, HugeInteger hugeInt2)**

**{**

**// return true if the value represented by**

**// elements of hugeInt1.intArray is not equal to**

**// value represented by elements of hughInt2.intArray**

**return true;**

**}**

**public static boolean isGreaterThan(HugeInteger hugeInt1, HugeInteger hugeInt2)**

**{**

**// return true if the value represented by**

**// elements of hugeInt1.intArray is greater than**

**// value represented by elements of hughInt2.intArray**

**return true;**

**}**

**public static boolean isLessThan(HugeInteger hugeInt1, HugeInteger hugeInt2)**

**{**

**// return true if the value represented by**

**// elements of hugeInt1.intArray is less than**

**// value represented by elements of hughInt2.intArray**

**return true;**

**}**

**public static boolean isGreaterThanOrEqualTo(HugeInteger hugeInt1, HugeInteger hugeInt2)**

**{**

**// return true if the value represented by**

**// elements of hugeInt1.intArray is greater than or equal to**

**// value represented by elements of hughInt2.intArray**

**return true;**

**}**

**public static boolean isZero(HugeInteger hugeInt1 )**

**{**

**// return true if the value represented by**

**// elements of hugeInt1.intArray is 0**

**return true;**

**}**

**//method to return a string**

**public String toString()**

**{**

**String s = Arrays.toString(this.intArray);**

**// return string representation of this object**

**return s;**

**}**

**}**