Here are some questions that you should be able to answer for the final. This is not necessarily the format of the final exam.

Question 1: What is a constructor? What is the purpose of a constructor in a class?

Answer: A constructor is a special kind of method in a class that has the same name as the name of the class and that has no return type, not even *void.* A constructor is called with the *new* operator when an object is created. Its main purpose is to initialize the newly created object, but in fact, it can do anything that the programmer wants it to do.

Question 2: Variables in a class can be either instance variables or static variables. When you declare a variable, you have to decide whether to make it static or instance. Explain how you can decide, and give several examples.

Answer: When a class declares an instance variable, every object of that class gets its own copy of the instance variable. This means that the variable can have different values in different objects. For example, in a Rectangle class, where an object represents a rectangle drawn on the screen, instance variables could be used to store the position, size, and color of the rectangle. Then every Rectangle would have its **own** position, size, and color. Suppose that the color were stored in a static variable instead of in an instance variable. A static variable belongs to the class rather than to the object created with that class. This means that there would be only one color that applies to all Rectangle objects. Changing the value of that static variable would, presumably, change the color of every Rectangle that exists!

On the other hand, there are some cases where static variables are appropriate. For one thing, static methods can only use static variables, not instance variables. Static variables can also be used to store data that pertains to the whole class. For example, suppose you want to keep track of the number of Rectangle objects that exist. This value can be stored in a static variable in the Rectangle class. It wouldn't make sense to use an instance variable since there is only one value, not one value per object.

Question 3: a) Suppose that you wanted to represent roman numbers as *objects.* How would you design a class, *RomanNumber,* so that each object belonging to the class represents a roman number? List the instance variables, methods, and constructors that you would include in the class, and state what each one is for. For the methods and constructors, specify the parameters and return type as well as the name; however, do **not** give full definitions. (The class you describe must work with part **b)** of this problem.)

b) Assume that the *RomanNumber* class from part **a)** is **already** written. Write a complete tester program, using that class, that will read two roman numbers from the user, add them, and print the result. The class you described in part **a)** must include any methods that you need to write this program.

Answer: (A Roman number has a value. The class needs an instance variable to represent that value. It is convenient to be able to construct Roman numbers from strings or from ints. Instance methods provide for the conversions in the opposite direction: from Roman number to int and from Roman number to String. This gives a pretty minimal class, but it's sufficient for writing the program in part **b**.)

```
a) class RomanNumber {
    int value;  // The int with the same numerical value as  this roman number.
     RomanNumber(String rn) { ... }   // Assume rn is a string such as "MCXII".
     RomanNumber(int n) { ... }       // Construct a roman number with value N.
     int getIntValue() { ... }   // Return the int with the same value as this roman number.
     String toString() { ... }  // Return a string representation of this Roman number.
}

B) public class AddRoman {
          public static void main (String[] args) {

             RomanNumber rn1 = new RomanNumber( roman1 );
             RomanNumber rn2 = new RomanNumber( roman2) );
             int sum = rn1.getIntValue() + rn2.getIntValue();
             RomanNumber sum = new RomanNumber( sum );
             System.out.println( "The sum is: " + sum.toString() );
          }
       }
```

Question 4: Write a method that finds the sum of all the numbers in an array of int's. The method should be named *arraySum.* It should have one parameter of type int[], and it should return a value of type int. The value that is returned should be obtained by adding up all the numbers in the array.

Answer:

```
public int arraySum(int[] a) {
  // Returns the sum of the numbers in a.
  int sum = 0;
  for (int i = 0; i < a.length; i++) {
    sum += a[i];
  }
  return sum;
}
```

Question 5: Define the following terms, as they relate to this course:

a) Reference to an object

Answer: A reference to an object is just the address of that object in memory. A variable can never hold an object, only a reference to an object. The object itself is stored on the heap. A reference acts as a "pointer" to the object, which can be used to find the object when necessary.

b) Subclass

Answer: A subclass is a class that extends (or is based on) another class, which is called its superclass. The subclass inherits all the behaviors and properties of the class. It can then add to or modify the inherited behaviors.

c) Polymorphism

Answer: Polymorphism refers to the fact that different objects can respond to the same method in different ways, depending on the actual type of the object. This can occur because a method can be overridden in a subclass. In that case, objects belonging to the subclass will respond to the method differently from objects belonging to the superclass.

(Note: If B is a subclass of A, then a variable of type A can refer to either an object of type A or an object of type B. Let's say that var is such a variable and that action() is a method in class A that is redefined in class B. Consider the statement "var.action()". Does this execute the method from class A or the method from class B? The answer is that there is no way to tell! The answer depends on what type of object var refers to, a class A object or a class B object. The method executed by var.action() depends on the actual type of the object that var refers to, not on the type of the variable var. This is the real meaning of polymorphism.)

d) this (when used in a method in a Java program)

Answer: When used in the definition of an instance method, this refers to the object to which the method belongs. Think of calling an instance method as sending a message to an object. Then this refers to the object that received the message.

Question 6: What is the relationship between "classes" and "objects"? What is the difference between them?

Answer: A class is used as a template for making objects. That is, a class contains information about what sort of data an object should contain and what sort of behaviors an object should have. One class can be used to make many objects. All the objects have the same behavior, as specified by the class, and they all hold the same kind of instance variables (although the data stored in those instance variables can be different from one object to another).

Classes and objects are totally different sorts of things! Objects do not exist until they are created, after a program is running, and they are destroyed before the program ends. A class is actually part of the program, so of course it exists the whole time a program is running. If you have a class, you can create a subclass based on that class, but you can't do anything similar with an object.

Question 7: Write a method named middle with three parameters of type int. The method should return the middle number among its three parameters, in order of size. For example, middle(3,17,12) would have the value 12; middle(737,-33,1266) would have the value 737; and middle(42,42,42) would have the value 42.

Answer: There are many ways to write this method. Here is the shortest:

```
     public static int middle(int x, int y, int z) {
    if ( x <= y && y <= z  ||  z <= y && y <= x )
      return y;  // y is in the middle
    if ( y <= x && x <= z  ||  z <= x && x <= y )
      return x;  // x is in the middle
    return z;  // it must be z that is in the middle
  }
```

You might find it easier to think about it this way: There are six different orders in which x, y, and z can occur. You can go through and check each possible order as one case in an if statement. So here is another way to write the method:

```
public static int middle(int x, int y, int z) {
    if ( x <= y && y <= z )
      return y;
    else if ( x <= z && z <= y )
      return z;
    else if ( y <= x && x <= z )
      return x;
    else if ( y <= z && z <= x )
      return z;
    else if ( z <= x && x <= y )
      return x;
    else // This is the case where z <= y && y <= x
      return z;
}
```

(It might be worth noting that Java won't let you use else if on the last line instead of just else. The problem is that with else if, it looks like there is a possibility that this routine won't return any value at all, and Java won't let you write such a routine. Of course, you know that you've covered all the possibilities and that in fact some value will always be returned. But Java isn't smart enough to figure that out.)

Question 8: Suppose that you want a very simple class to represent the money in a bank account. It needs three methods, one to deposit a given amount into the account, one to withdraw a given amount, and one to check how much money is in the account. It also needs a constructor that creates an account containing a specified initial amount of money. The class should have one instance variable, for storing the amount of money in the account. Write a complete Java class satisfying this requirement.

Answer:

```
public class BankAccount {
  protected double balance;   // amount of money in the account

  public BankAccount(double initialDeposit) {  // constructor
    balance = initialDeposit;
  }

  public void deposit(double amount) {
    balance = balance + amount;
  }

  public void withdraw(double amount) {
    balance = balance - amount;
  }

  public double checkBalance() {
    return balance;
  }
}
```

Question 9: Suppose that you are given a class named Sorter. An object of this class keeps a sorted list of numbers. When a new number is added, it is inserted into its correct place in the list. For example, if 17.42 is added to the list 3.4, 12.1, 19.96, 20.0, then the list will be modified to 3.4, 12.1, 17.42, 19.96, 20.0. The constructor in the Sorter class creates an initially empty list. The class includes the methods:

```
void add(double newNum)  -- adds a number to the list
double get(int N)  -- returns the Nth number from the list
```

Write a complete Java program that uses a Sorter object to sort a list of 10 numbers entered by the user. The program should ask the user to enter 10 numbers. It should read each number and add it to the Sorter object. Then it should retrieve the numbers from the Sorter object in order and print out each number on the console.

Answer:

```
public class TestSorter  {

    public  static void main  (String[] args) {

       int x = 50, y = 75;
      Sorter sort = new Sorter();   // create a Sorter object

      for (int i = 0; i < 9; i++) {
         double num = Math.random() * 100 + 1
         sort.add(num);                     //   ... add the number to the Sorter
      }

     System.out.println("The numbers in sorted order are: ");

      for (int i = 0; i < 9; i++) {
         double num = sort.get(i);    // get the i-th number from the sorter...
        System.out.println(""+num, );           //   ...and print it out
         y+= 20;
      }
    }
  }  // end of class TestSorter
```

Question 10: Recall that rolling a die can be simulated with the command

```
die = (int)(6 * Math.random()) + 1;
```

Why does this work? What is Math.random()*?* Why is it multiplied by 6? What is the effect the (int) and why is it necessary?

Answer: Math.random() is a built-in method in Java. Each time it is called, it returns a random real number that is >= 0.0 and < 1.0. Multiplying the random number by 6 gives a number that is >= 0.0 and strictly < 6.0. Since the number on a die is an integer, it is necessary to throw away the fractional part of the number. This is done by the "type cast" operation, (int), which takes a real number and turns it into an integer by dropping any digits that follow the decimal point. In this case, the result is one of the integers 0, 1, 2, 3, 4, or 5. Finally, adding 1 gives an integer in the range 1 to 6, which is just what is needed to represent the roll of a die.

Question 11: Suppose that income tax is determined as follows: If income is less than $10,000, then there is no tax. For incomes between $10,000 and $30,000, the tax is 0.15 times (income - 10000). For incomes greater than $30,000, the tax is 3000 plus 0.3 times (income - 30000). Write a program segment that inputs the amount of income from the user and prints out the income tax on that amount. You do not have to write a complete program, but you should declare any variables that you use.

Answer: There are, of course, several ways to do this. Here is one example:

```
double income;  // amount of income, to be input by the user
double tax;     // amount of tax on the specified income
// assume some input value into variable income
if (income < 10000) {
  System.out.println("There is no tax on an income of $" + income);
} else if (income < 30000) {
  tax = 0.15 * (income - 10000);
  System.out.println("The tax on $" + income + " is $" + tax);
} else {
  tax = 3000 + 0.3 * (income - 30000);
  System.out.println("The tax on $" + income + " is $" + tax);
}
```

Question 12: Write a program segment that simulates rolling a pair of dice over and over until a pair of threes is rolled (that is, until the number on each die is equal to 3). The program should count the number of rolls and output this number at the end. You do not have to write a complete program, but you should declare any variables that you use. (The formula for rolling a die is given in problem **10**.)

Answer: Here are two possible answers. Note that "die1 == 3 && die2 == 3" is the condition for ending the loop, since it says that both dice show 3. The condition for continuing the loop is the opposite of this, "die1 != 3 || die2 != 3".

Version 1:

```
int die1, die2;  // values showing on dice
int rollCt = 0;  // number of times dice have been rolled
while (true) {
  die1 = (int)(6 * Math.random()) + 1;
  die2 = (int)(6 * Math.random()) + 1;
  rollCt++;
  if (die1 == 3 && die2 == 3)
     break;
}
System.out.println("Number of rolls = " + rollCt);
```

Version 2:

```
int die1 = 0;   // values on dice; initial values of zero
int die2 = 0;   //     are required to make the while loop
                //     execute the first time
int rollCt = 0; // number of times dice have been rolled
while (die1 != 3 || die2 != 3) {
  die1 = (int)(6 * Math.random()) + 1;
  die2 = (int)(6 * Math.random()) + 1;
  rollCt++;
}
System.out.println("Number of rolls = " + rollCt);
```

Question 13: Show the exact output produced by the following Java program. (Explain your work, to get partial credit.)

```
public class TestQuestion {
  public  static void main(String[] args) {
    int a, b, c,;
    a = 30;
    b = 2;
    c = 0;
    while ( a > b ) {
      System.out.println ("b is " + b);
      b = (a + b) / 2;
      c++;
      a--;
    }
    System.out.println("c is " + c);
  }
}
```

Answer: The exact output of the program consists of the following lines written:

```
b is 2
b is 16
b is 22
b is 25
c is 4
```

Each time through the while loop, the current value of b is output by the first line in the loop. The first time through, the value of b is 2. The value of b then changes to (a+b)/2, which is (30+2)/2, or 16. The value of c changes to 1, and the value of a changes to 29.

The second time through the loop, the value of b that is output to the console is 16. Then the value changes to (a+b)/2, which is now (29+16)/2, or 45/2, or 22. Note that the answer is 22, not 22.5, because when two integers are divided in Java, the result is defined to be an integer. Essentially, the fractional part is thrown away. In this iteration of the loop, c changes to 2 and a changes to 28.

This continues for two more steps, outputting 22 and 25 as the values of b. At this point, b becomes 26, c becomes 4, and a becomes 26. Since a is no longer > b, the loop ends and the final value of c is output.

Question 14: Write a complete Java method named pythagorus. It should have two parameters, x and y, of type double. It should compute and return as its value: the square root of ($x^2 + y^2$).

Answer:

```
public double pythagorus(double x, double y) {
   return Math.sqrt( x*x + y*y );
}
```

You could also do

```
      return Math.sqrt(add Math.pow(x,2) + Math.pow(y,2))l
```

Question 15: Show the exact output produced by the following program:

```
class Example  {
    public static void main (String[] args) {
      int a = 1, b = 10;
      do {
        int z = a*b;
        System.out.println(""+z);
        a++;
        b--;
      } while (a < b);
    }
 } // end of program Example
```

Answer: Each execution of the loop produces a line of output containing one number. The first time through the loop, a is 1, b is 10, and so z is 10. Then a is incremented to 2, b is decremented to 9, and the loop repeats. This time, the output is 2*9, or 18. This continues until x becomes 6 and y becomes 5. So the output is:

```
10
18
24
28
30
```

Question 16 Write a for statement that will compute the sum of the first 100 integers, that is, 1*1 + 2*2 + 3*3 + ... + 100*100. You should **not** write a complete program, but you **should** declare any variables that you use.

Answer:

```
int sum = 0;  // the sum of all the squares
for (int n = 1; n <= 0; n++)
  sum += n*n;
```

Question 17: If str is a variable of type String, then str.length() is the length of the string (that is, the number of characters that it contains), and that str.charAt(N) is the character at position N in the string (where positions are numbered starting from zero). Write a Java program segment that will read in a string typed by the user and will print out a reversed copy of that string. For example, if the user types the string "Hello Java", then the computer should respond with "avaJ olleH".

Answer: Note that if length is the length of a string, then the characters in that string are numbered 0,1,2,...,length-1. The last character is in position number length-1, not in position number length.

```
String answer;  // string typed in by the user
System.out.println("Please enter a string: ");
// get the input from the user
int length = answer.length();
for (int N = length-1;  N >= 0; N--) {
  System.out.println(answer.charAt(N));
}
```

Question 18: Write a class called Square. In this class you will have a constructor that will take in the width of the box as a float OR the height as an integer. The width will always be half the size of the height.. You must be able to change the width and the height, as well as, be able to ask for the width and the height. Finally, you are to write a method that will take in a Square object and return back the combined width times the combined height of the two squares as an integer.

Here is a driver class for you and the output:

```
public class Driver {
  public static void main(String[] args) {
    Square s1 = new Square(11);
    Square s2 = new Square(3.25f);

    System.out.println("Square ones height: " + s1.getHeight() + " and width: " + s1.getWidth());
    System.out.println("Square twos height: " + s2.getHeight() + " and width: " + s2.getWidth());
    s2.setWidth(4.5f);System.out.println("Square twos new height: " + s2.getHeight() + " and width: " + s2.getWidth());
    System.out.println("The combined height and width: " + s1.combinedSize(s2));
  }
}
```

OUTPUT:

```
Square ones height: 11 and width: 5.5
Square twos height: 6 and width: 3.25
Square twos new height: 9 and width: 4.5
The combined height and width: 200
```

Answer:

```
public class Square {
private int height;
private float width;

public Square(int height) {
  this.height = height;
  this.width = height / 2.0f;
}
public Square(float width) {
  this.height = (int) (width * 2);
  this.width = width;
}
public int getHeight() {
  return height;
}
public void setHeight(int height) {
  this.height = height;
  this.width = height / 2.0f;
}
public float getWidth() {
  return width;
}
public void setWidth(float width) {
  this.width = width;
  this.height = (int) (width * 2);
}
public int combinedSize(Square s){
  float cWidth = width + s.getWidth();
  int cLength = height + s.getHeight();
  return (int)(cWidth * cLength);
}
}
```

Question 19: Write a complete and syntactically correct Java program that outputs the following message in the console window:

The even numbers between 2 and 98

and then outputs the specified even numbers using a while loop that will only display 5 even numbers per line.

Answer:

```
public class PrintEvenNumbers  {
  public static void main(String[] args) {
    int k = 2;
    System.out.println("The even numbers between 2 and 98 are \n");
    while (k <= 98)  {
      if (k % 5 == 0 ) {
        System.out.println(k);
      } else  {
        System.out.print( k + "\t");
      }
      k = k + 2;
    }
    System.out.println("\n");
  }
}
```

Question 20: Write a complete program that outputs the following:

```
--->
   --->
      --->
         --->
         <---
      <---
   <---
<---
```

Answer:

```
public class Arrows  {
  public static void main(String[] args) {
    int j = 0;
    String t = "";
    while (j < 4) {
     System.out.println(t + " ---> ");
     j = j + 1;
     t += "\t";
    }

    while (j > 0) {
     t = "";
     for ( int i = 0; i <j-1; i++)
        t += "\t";

     System.out.println (t + " <---");
     j = j - 1;
    }
  }
}
```

Question 21: Write a syntactically correct Java class **Triangle** which stores the three sides of a triangle (i.e., a, b, and c which are all of type double) as object variables. Triangle has a constructor which is used to initialize the three sides of the triangle and a method which returns the perimeter of the triangle.

Answer:

```
public class Triangle  {
  private double a, b, c;
  public double peri;

  public void Triangle(double ka, double kb, double kc) {
    a = ka; b = kb; c = kc;
    peri =  perimeter(a,b,c);
  }

  public double perimeter(double a, double b, double c) {
   return a + b + i;
  }
}

public class TriangleTester  {
  public static  void main (String[] args)) {
    Triange t = new  Triangle(2.0, 3.0, 4.0);
    System.out.println (t.peri);
   }
}
```

Question 22: What is meant by the term "object instantiation"?

Answer: Object are instantiated from classes. Instantiating an object means creating an object and allocating storage for it and potentially initializing it. One can create any number of objects from a class.

Question 23: What are the kinds of comments in the programming language Java?

Answer: /* comment */ OR   // comment

Question 24: What is meant by the term "operator precedence"?

Answer: Operators are evaluated according to the precedence table which defines the precedence level and associatively for each operator.

Question 25: List a Java (i) keyword, an (ii) identifier, and a (iii) literal.

Answer: (i) class; (ii) k; (iii) -17.34

Question 26: Write a syntactically correct Java declaration of a read-only variable where the variable name is E, the type is double, and the value is 2.81.

Answer: final double E = 2.81;

Question 27: Write a syntactically correct infinite while loop in Java.

Answer: while (true) { ... }

Question 28: Find the divide by zero error in the following program and modify the program so that we display a message if we are about to divide by zero.

```
public class BadDivide {
  public static void main(String[] args)  {
    int result = 0;
    int n = 100;
    for (int d = 5; d >= 0; d--)  {
      System.out.println("result = " + n / d);
    }
  }
}
```

Answer:

```
public class BadDivide {
 public static void main(String[] args)  {
   int result = 0;
   int n = 100;
   for (int d = 5; d >= 0; d--)  {
     if ( d == 0)
       System.out.println("Error: divide by zero");
     else
      System.out.println("result = " + n / d);
    }
  }
}
```

Question 29: For the following, write code segments that will perform the specified action. Assume that all variables have already been declared and given values.

a. Print "Hurrah!" if sum is evenly divisible by count.

```
if (sum%count == 0)
   System.out.println ("Hurrah!");
```

b. Increment the integer variable total if total is zero and decrement total otherwise.

```
if (total == 0)
  total++;
else
  total--;
```

c. Print "num is zero", "num is negative", or "num is positive" as appropriate based on the current value of num.

```
if (num == 0)
  System.out.println ("num is zero");
else if (num < 0)
  System.out.println ("num is negative");
else
  System.out.println ("num is positive");
```

d. Print "num is zero", "num is even", or "num is odd" as appropriate based on the current value of num.

```
if (num == 0)
  System.out.println ("num is zero");
else if (num%2 == 0)
  System.out.println ("num is even");
else
  System.out.println ("num is odd");
```

e. Assign the smallest of two integer values num1 and num2 to the variable smallest. (use an if-else statement)

```
if (num1 < num2)
  smallest = num1;
else
  smallest = num2;
```

f. Assign the smallest of three integer values num1, num2, and num3 to the variable smallest. (do not use logical operators)

```
if (num1 < num2)
  if (num1 < num3)
    smallest = num1;
  else
    smallest = num3;
else
  if (num2 < num3)
    smallest = num2;
  else
    smallest = num3;
```

g. Assign the smallest of three integer values num1, num2, and num3 to the variable smallest. (use logical operators)

```
if (num1 < num2 && num1 < num3)
  smallest = num1;
else if (num2 < num3)
  smallest = num2;
else
  smallest = num3;
```

h. Print "Uppercase", "Lowercase", or "Not a letter" depending on whether the character stored in ch is an uppercase alphabetic character, a lowercase alphabetic character, or not an alphabetic character at all.

```
if (ch >= 'A' && ch <= 'Z')
 System.out.println ("Uppercase");
else if (ch >= 'a' && ch <= 'z')
 System.out.println ("Lowercase");
else
  System.out.println ("Not a letter");
```

Question 30: Write code segments that will perform the specified action for the following questions:

a. Print the odd numbers between 1 and 100.

```
for (int num=1; num <= 99; num += 2)
 System.out.println(num);
```

b. Print the multiples of 3 from 300 down to 3.

```
for (int num=300; num >= 3; num-=3)
 System.out.println(num);
```

c. Print the numbers between LOW and HIGH that are evenly divisible by four but not by five.

```
for (int count=LOW; count <= HIGH; count++)
  if (count%4 == 0 && count%5 != 0)
   System.out.println (count);
```

d. Determine and print the number of times the character 'a' appears in the String variable str.

```
count = 0;
for (int index=0; index < str.length(); index++)
  if (str.charAt(index) == 'a')
    count++;
System.out.println ("Number of a's: " + count);
```

Question 31: For each, write the method described. Assume all ranges are inclusive (include both end points).

a. Write a method called powersOfTwo that prints the first 10 powers of 2 (starting with 2). The method takes no parameters and doesn't return anything.

```
public void powersOfTwo () {
  int power =1;
  for (int count=1; count <= 10; count++) {
    power *= 2;
   System.out.println(power);
  }
}
```

b. Write a method called alarm that prints the word “Alarm!” multiple times on separate lines. The method should accept an integer parameter that specifies how many times the output line is printed.

```
public void alarm (int num){
  for (int count=1; count <= num; count++) {
   System.out.println("Alarm");
 }
}
```

c. Write a method called sum100 that returns the sum of the integers from 1 to 100.

```
public int sum100 () {
  int sum = 0;
  for (int count=1; count <= 100; count++)
    sum += count;
  return sum;
}
```

d. Write a method called countA that accepts a String parameter and returns the number of times the letter 'A' is found in the string.

```
public int countA (String str) {
  int count = 0;
  for (int index=0; index < str.length(); index++)
    if (str.charAt(index) == 'A')
      count++;
  return count;
}
```

e. Write a method called multiConcat that takes a String and an integer as parameters, and returns a String that is the parameter string concatenated with itself n number of times (where n is the second parameter). For example, if the parameters are "hi" and 4, the return value is "hihihihi".

```
public String multiConcat (String str, int max) {
  String result = "";
  for (int count=1; count <= max; count++)
    result += str;
  return result;
}
```

f. Write a method called randomInRange that accepts two integer parameters representing a range. You may assume that the first parameter is less than or equal to the second, and that both are positive. The method should return a random integer in the specified range.

```
public int randomInRange (int low, int high) {
  int x = ((int) (Math.random()*(high-(low-1))) + low);
  return x;
}
```

g. Overload the randomInRange method of the previous exercise such that if only one parameter is provided, the range is assumed to be from 1 to that value. You may assume the parameter value is positive.

```
public int randomInRange (int high){
  int x = ((int) (Math.random()*high) + 1);
  return x;
}
```

Question 31: Inherit the following class Animal to create a new class Cat. You must write an adequate constructor and a method show() that will display the cats name and breed of the cat.

```
public class AnimalTest {
 public static void main(String[] args)  {
    Animal aCat;                  // Declare an animal variable
    aCat = new Cat("Fang", "mixed");
    aCat.show();
  }
}

class Animal  {
  private String type;

  public Animal(String aType) {
    type = new String(aType);
  }

  public void show() {
    System.out.println("This is a " + type);
  }
}
```

Answer:

```
class Cat extends Animal {
 private String name;     // Name of a cat
 private String breed;    // cat breed

 public Cat(String aName, String aBreed) {
   super("Cat");        // Call the base constructor
   name = aName;        // Supplied name
   breed = aBreed;      // Supplied breed
 }

 public void show() {
   super.show();        // Call the base method
   System.out.println("It's " + name + " the " + breed + " breed");
 }
}
```