Program for Newton Raphson Method

Given a function f(x) on floating number x and an initial guess for root, find root of function in interval. Here f(x) represents algebraic or transcendental equation.

For simplicity, we have assumed that derivative of function is also provided as input.

**Example:**

Input: A function of x (for example x3 – x2 + 2),

derivative function of x (3x2 – 2x for above example)

and an initial guess x0 = -20

Output: The value of root is : -1.00

OR any other value close to root.

We have discussed below methods to find root in set 1 and set 2

Bisection method,

Method of false position method

**Comparison with above two methods:**

1. In previous methods, we were given an interval. Here we are required an initial guess value of root.
2. The previous two methods are guaranteed to converge, Newton Rahhson may not converge in some cases.
3. Newton Raphson method requires derivative. Some functions may be difficult to  
   impossible to differentiate.
4. For many problems, Newton Raphson method converges faster than the above two methods.
5. Also, it can identify repeated roots, since it does not look for changes in the sign of f(x) explicitly

**The formula:**  
Starting from initial guess x1, the Newton Raphson method uses below formula to find next value of x, i.e., xn+1 from previous value xn.  
[newtonraphsonformula](https://media.geeksforgeeks.org/wp-content/cdn-uploads/newtonraphsonformula.png)

**Algorithm:**  
Input: initial x, func(x), derivFunc(x)  
Output: Root of Func()

1. Compute values of func(x) and derivFunc(x) for given initial x
2. Compute h: h = func(x) / derivFunc(x)
3. While h is greater than allowed error ε
   1. h = func(x) / derivFunc(x)
   2. x = x – h

============================================

class GFG {

    static final double EPSILON = 0.001;

    // An example function whose solution

    // is determined using Bisection Method.

    // The function is x^3 - x^2 + 2

    static double func(double x)

    {

        return x \* x \* x - x \* x + 2;

    }

    // Derivative of the above function

    // which is 3\*x^x - 2\*x

    static double derivFunc(double x)

    {

        return 3 \* x \* x - 2 \* x;

    }

    // Function to find the root

    static void newtonRaphson(double x)

    {

        double h = func(x) / derivFunc(x);

        while (Math.abs(h) >= EPSILON)

        {

            h = func(x) / derivFunc(x);

            // x(i+1) = x(i) - f(x) / f'(x)

            x = x - h;

        }

        System.out.print("The value of the"

                + " root is : "

                + Math.round(x \* 100.0) / 100.0);

    }

    public static void main (String[] args)

    {

        // Initial values assumed

        double x0 = -20;

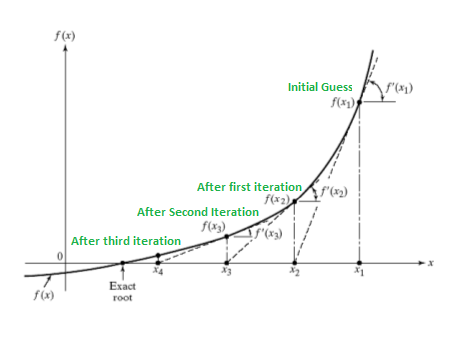
        newtonRaphson(x0);

    }

}

**Output:**

The value of root is : -1.00

**How does this work?**  
The idea is to draw a line tangent to f(x) at point x1. The point where the tangent line crosses the x axis should be a better estimate of the root than x1. Call this point x2. Calculate f(x2), and draw a line tangent at x2.  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/newtonRaphsonMethod.png)

We know that slope of line from (x1, f(x1)) to (x2, 0) is f'(x1)) where f’ represents derivative of f.