```
public class Prime {



    private int sizeLimit;
    private int size;
    private int value;
    public Prime(int _size, int size_limit){
        this.setSizeLimit(size_limit);
        this.setSize(_size);
        ArrayList<Integer> primes = new ArrayList<Integer>();
        for (int i = 3; i < Math.pow(10, sizeLimit)){
            boolean isPrime = true;
            for (int j=2; j<i ;j++){
                if ((i % j)==0){
                    isPrime=false;
                    break;
                }
            }
            if (isPrime){
                primes.add(i);
            }
        }
        ArrayList<Integer>desirablePrimes = new ArrayList<Integer>();
        for (int i=0; i < primes.size(); i++){
            Integer prime = primes.get(i);
            if (String.valueOf(prime).length() == size){
               desirablePrimes.add(prime);
            }
            else if (String.valueOf(prime).length() > size){
                break;
            }
        }
        Random prime_getRandom = new Random();
        int random = prime_getRandom.nextInt(desirablePrimes.size());
        this.value = desirablePrimes.get(random);
        }
        public prime(int size){
           this(size, size+1);
        }
        public prime(){
            this(10, 11);
        }





    }
/**
When this class is calculating primes, it needs to stop at some point because the algorithm
and computing power would eventually take inordinate amounts of time to compute. In this
method, you can set the limit on the number of digits a prime can have. You can set the limit
as high as 40 (as of this writing), which is mostly arbitrary; I may update it. If you
attempt to set it above 40, it defaults to the limit (40 as of this writing).
@param size_limit the length, in digits, that the class should be allowed to calculate. If
you set the limit lower, it will compute primes faster, but they cannot be as long. If you
set the limit higher, they will be stronger and lead to stronger encryption, but the program
will take more time. Do not exceed (as of this writing) 40; otherwise, the limit will simply
be capped at 40, no more.
@return whether you have passed in a valid limit, which is the same thing as whether your
desired limit has been set. Returns false if you tried to set a limit over 40, signalling
that it has defaulted to a limit of 40.
*/
```