```
/*@ pure @*/ public class Complex {
private double re, im;
//CREATORs
//@ensures this.re()==0 && this.im()==0;
public Complex() { this.re = 0; this.im = 0; }

//@ensures this.re()==re && this.im()==im;
public Complex (double re, double im) {this.re = re; this.im = im; }

//OBSERVERs
//@ensures (*\result == valore parte reale di this*);
public double re() {return this.re;}

//@ensures (*\result == valore parte immaginaria di this*);
public double im() {return this.im;}

//@ensures \result == Math.sqrt(this.re()*this.re()+this.im()*this.im());
public double rho() { return Math.sqrt(re*re+im*im); }

//@ensures (*\result == valore fase di this in radianti*)  && 0 <=
Math.abs(\result) <= Math.PI;
public double phi() {
  if (re==0 && im==0) return 0;
  double f = Math.atan(im/re);
  return im >= 0 ? f : -f;
}

//PRODUCERs

//@ensures this.re()==c.re() && this.im()==c.im();
public Complex (Complex c) { this.re = c.re(); this.im = c.im();}

/*@ requires c != null;
   @ ensures (\result.re()==this.re()+c.re()) && (\result.im()==this.im()
+c.im());
   @*/
public Complex sum(Complex c) {  return new Complex(re+c.re,im+c.im); }

/*@ requires c != null;
   @ ensures (\result.re()==this.re()-c.re()) && (\result.im()==this.im()-
c.im());
   @*/
public Complex sub(Complex c) {  return new Complex(re-c.re, im-c.im); }

/*@ requires c != null;
  @ ensures (\result.rho()==this.rho()*c.rho()) &&
  @         (\result.phi()==this.phi()+c.phi());
  @*/
public Complex mul(Complex c) {return new Complex(re*c.re-im*c.im,
re*c.im+im*c.re); }

/*@ requires c != null;
  @ ensures c.rho()!=0 &&  (\result.rho()==this.rho()/c.rho()) &&
  @     (\result.phi()==this.phi()-c.phi());
  @ signals(DivideByZeroException e) c.rho()==0;
  @*/
public Complex div(Complex c) throws DivideByZeroException {
  double rhoQuadro =  c.re*c.re + c.im*c.im;
  if (rhoQuadro==0) throw new DivideByZeroException();
  return new Complex( (re*c.re+im*c.im)/rhoQuadro ,
      (c.re*im - re*c.im) / rhoQuadro);
}
```

```
//@ ensures \result.re() == c.re() && \result.im()== - c.im();
public Complex conjugate() {  return new Complex(c.re, -c.im); }

//no mutators: pure class

// UTILITY
//@ensures \result.re()==0&&\result.im()==-1;
public static Complex j() { return new Complex(0, 1); }

//toString: IMPLEMENTAZIONE DELL'AF
public String toString() {  return re + " + " + im + "j"; }

//ridefire "equals" quando l'ADT è IMMUTABILE
public boolean equals(Complex c) {  return re == c.re && im == c.im; }

public boolean equals(Object o) {
  if (!(o instanceof Complex)) return false;
  return equals((Complex)o);
}

} // end class
```