Task1: Key

```
import java.util.Scanner;

public class Task1 {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Enter the first complex number: ");
      double a = input.nextDouble();
      double b = input.nextDouble();
      Complex c1 = new Complex(a, b);

      System.out.print("Enter the second complex number: ");
      double c = input.nextDouble();
      double d = input.nextDouble();
      Complex c2 = new Complex(c, d);

      System.out.println("(" + c1 + ")" + " + " + "(" + c2 + ")" + " = " + c1.add(c2));
      System.out.println("(" + c1 + ")" + " - " + "(" + c2 + ")" + " = " + c1.subtract(c2));
      System.out.println("(" + c1 + ")" + " * " + "(" + c2 + ")" + " = " + c1.multiply(c2));
      System.out.println("(" + c1 + ")" + " / " + "(" + c2 + ")" + " = " + c1.divide(c2));
      System.out.println("|" + c1 + "| = " + c1.abs());

      Complex c3 = (Complex)c1.clone();
      System.out.println(c1 == c3);
      System.out.println(c3.getRealPart());
      System.out.println(c3.getImaginaryPart());
   }
}

class Complex implements Cloneable {
   private double a = 0, b = 0;

   public Complex() {    }

   Complex(double a, double b) {
      this.a = a;
      this.b = b;
   }

   public Complex(double a) {       this.a = a;    }

   public double getA() {       return a;    }

   public double getB() {       return b;    }

   public Complex add(Complex secondComplex) {
      double newA = a + secondComplex.getA();
      double newB = b + secondComplex.getB();
      return new Complex(newA, newB);
   }

   public Complex subtract(Complex secondComplex) {
      double newA = a - secondComplex.getA();
      double newB = b - secondComplex.getB();
      return new Complex(newA, newB);
   }
```

```
  public Complex multiply(Complex secondComplex) {
    double newA = a * secondComplex.getA() - b * secondComplex.getB();
    double newB = b * secondComplex.getA() + a * secondComplex.getB();
    return new Complex(newA, newB);
  }

  public Complex divide(Complex secondComplex) {
    double newA = (a * secondComplex.getA() + b * secondComplex.getB())
        / (Math.pow(secondComplex.getA(), 2.0) + Math.pow(secondComplex.getB(),
        2.0));
    double newB = (b * secondComplex.getA() - a * secondComplex.getB())
        / (Math.pow(secondComplex.getA(), 2.0) + Math.pow(secondComplex.getB(),
        2.0));
    return new Complex(newA, newB);
  }

  public double abs() {
    return Math.sqrt(a * a + b * b);
  }

  @Override
  public String toString() {
    if (b != 0)
      return a + " + " + b + "i";
    return a + "";
  }

  public double getRealPart() {       return a;    }

  public double getImaginaryPart() {       return b;    }

  @Override /** Implement the clone method in  the Object class */
  public Object clone() {
    try {
      return super.clone(); // Automatically perform a shallow copy
    }
    catch (CloneNotSupportedException ex) {
      return null;
    }
  }
}
```

Task2: Key

```
public interface MovieInterface {
}
```

```
public interface ComedyInterface {
}
```

```
public interface ActionInterface {
}
```

```
public interface RomanceInterface {
}
```

```
public interface MysteryInterface {
}
```

```
import java.util.Arrays;

public abstract class Movie implements MovieInterface, Comparable<Movie>, Cloneable{
   private String type;
   private String title;
   private String[] starring;
   private int runningTime;
   private String country;
   private String language;

   public Movie(String title, String[] starring, int runningTime, String country, String language) {
      this.title = title;
      this.starring = starring;
      this.runningTime = runningTime;
      this.country = country;
      this.language = language;
   }

   public String getType() {       return type;    }

   public String getTitle() {       return title;    }

   public String[] getStarring() {       return starring;    }

   public int getRunningTime() {       return runningTime;    }

   public String getCountry() {       return country;    }

   public String getLanguage() {       return language;    }

   public void setTitle(String title) {       this.title = title;    }
```

```
    public void setStarring(String[] starring) {       this.starring = starring;    }

    public void setCountry(String country) {       this.country = country;    }

    public void setRunningTime(int runningTime) {       this.runningTime = runningTime;    }

    public void setLanguage(String language) {       this.language = language;    }

    public void setType(String type) {       this.type = type;    }

    @Override
    public abstract String toString() ;

    public String MovieInfo() {
        return " Movie{" +
                "title=" + title +
                ", starring=" + Arrays.toString(starring) +
                ", runningTime=" + runningTime +
                ", country=" + country +
                ", language=" + language + '}';
    }

    @Override
    public int compareTo(Movie m) {
        int typeDiff = type.compareTo(m.getType());
        if(typeDiff!=0)
            return typeDiff;
        return runningTime-m.getRunningTime();
    }
}
```

```
public class Mystery extends Movie implements MysteryInterface {

    public Mystery(String type, String title, String[] starring, int runningTime, String country, String language) {
        super(title, starring, runningTime, country, language);
        setType(type);
    }

    @Override
    public String toString() {
        return "Mystery"+ MovieInfo();
    }
}
```

```
public class Rescue extends Movie implements ActionInterface, RomanceInterface {

    public Rescue(String type, String title, String[] starring, int runningTime, String country, String language) {
        super(title, starring, runningTime, country, language);
        setType(type);
    }

    @Override
    public String toString() {
        return "Rescue"+ MovieInfo();
```

```
    }
}
```

```
public class Romance extends Movie implements RomanceInterface {

    public Romance(String type, String title, String[] starring, int runningTime, String country, String language) {
        super(title, starring, runningTime, country, language);
        setType(type);
    }

    @Override
    public String toString() {
        return "Romance"+ MovieInfo();
    }
}
```

```
public class RomanticComedy extends Movie implements RomanceInterface, ComedyInterface {

    public RomanticComedy(String type, String title, String[] starring, int runningTime, String country, String language) {
        super(title, starring, runningTime, country, language);
        setType(type);
    }

    @Override
    public String toString() {
        return "RomanticComedy"+ MovieInfo();
    }
}
```

```
public class Action extends Movie implements ActionInterface {

    public Action(String type, String title, String[] starring, int runningTime, String country, String language) {
        super(title, starring, runningTime, country, language);
        setType(type);
    }

    @Override
    public String toString() {
        return "Action" + MovieInfo();
    }
}
```

```
public class Comedy extends Movie implements ComedyInterface {

    public Comedy(String type, String title, String[] starring, int runningTime, String country, String language) {
        super(title, starring, runningTime, country, language);
        setType(type);
    }
```

```
    @Override
    public String toString() {
        return "Comedy"+ MovieInfo();
    }
}
```

---

public class **HollywoodBlockbuster** extends **Movie** implements **ActionInterface, MysteryInterface, RomanceInterface, ComedyInterface** {

```
    public HollywoodBlockbuster(String type, String title, String[] starring, int runningTime, String country, String language) {
        super(title, starring, runningTime, country, language);
        setType(type);
    }

    @Override
    public String toString() {
        return "HollywoodBlockbuster"+ MovieInfo();
    }
}
```

```
import java.io.*;
import java.lang.reflect.Array;
import java.util.*;

public class Task2 {
    public static void main(String[] s) throws Exception{
        ArrayList<Movie> movies = new ArrayList<>();

        File f = new File("movies.txt");
        if(f.exists()){
            Scanner in = new Scanner(f);
            in.nextLine();  // ignore first line
            while(in.hasNext()){
                String l = in.nextLine().trim();
                switch (getType(l)){
                    case "Action":
                        movies.add(new Action(getType(l), getTitle(l), getStarring(l), getRunningTime(l),  getCountry(l), getLanguage(l)));
                        break;
                    case "Romance":
                        movies.add(new Romance(getType(l), getTitle(l), getStarring(l), getRunningTime(l),  getCountry(l), getLanguage(l)));
                        break;
                    case "Comedy":
                        movies.add(new Comedy(getType(l), getTitle(l), getStarring(l), getRunningTime(l),  getCountry(l), getLanguage(l)));
                        break;
                    case "Mystery":
                        movies.add(new Mystery(getType(l), getTitle(l), getStarring(l), getRunningTime(l),  getCountry(l), getLanguage(l)));
                        break;
                    case "Rescue":
                        movies.add(new Rescue(getType(l), getTitle(l), getStarring(l), getRunningTime(l),  getCountry(l), getLanguage(l)));
                        break;
                    case "Romantic Comedy":
```

```
                movies.add(new RomanticComedy(getType(l), getTitle(l), getStarring(l), getRunningTime(l),  getCountry(l),
getLanguage(l)));
                break;
             case "Hollywood Blockbuster":
                movies.add(new HollywoodBlockbuster(getType(l), getTitle(l), getStarring(l), getRunningTime(l),  getCountry(l),
getLanguage(l)));
                break;
          }
       }
       Collections.sort(movies);

       PrintWriter out = new PrintWriter("sorted_movies.txt");
       for (int i=0; i<movies.size();i++) {
          Movie m = movies.get(i);
          out.println(m.getType()+": "+ m.getTitle() + ','
                + Arrays.toString(m.getStarring()) +
                ", " + m.getRunningTime() +
                ", " + m.getCountry() +
                ", " + m.getLanguage()+".");
       }
       out.close();
    }
    else {
       System.out.println("Error: File not found!");
    }
  }

  public static String getType(String l){
    return l.split(":")[0].trim();
  }

  public static String getTitle(String l){
    return l.split("[:,]")[1].trim();
  }

  public static String[] getStarring(String l){
    return l.substring(l.indexOf('[')+1, l.indexOf(']')).trim().split(",");
  }

  public static int getRunningTime(String l){
    return Integer.parseInt(l.substring(l.indexOf(']')+2).trim().split(",")[0]  );
  }

  public static String getCountry(String l){
    return l.substring(l.indexOf(']')+2).trim().split(",")[1].trim() ;
  }

  public static String getLanguage(String l){
    return l.substring(l.indexOf(']')+2).trim().split("[,.]")[2].trim();
  }
}
```