Inheritance

```
import java.io.*;import java.lang.*;
class d2vectors { protected float i,j;
   public d2vectors(float a, float b){ i=a;  j=b;  }
   public float slope( ){return (j/i); }   public float magsqr( ){return (i*i+j*j);}
   public void dble( ){i=2*i;j=2*j;}    public void pt( ){System.out.print(i+"i+"+j+"j");}
} -> public void ab(d2vectors p){System.out.println(1000*i + 100*j +
                                                              10*p.i + p.j);}
class d3vectors extends d2vectors{float k;
   public d3vectors(float p, float q, float r){super( p,q);k=r; }
   public float magsqr( ) {return(super.magsqr( )+k*k);}
   public void ttt( ) {super.dble( );k=k*3;} //Remove super
   public void dble( ){i=2*i;j=2*j;k=2*k;}
   public double direction( ){return (k/Math.sqrt(i*i+j*j));}
   // public d2vectors project( ){return super;} super can not be used like keyword "this"
   public void pt( ){System.out.print(i+"i+"+j+"j+"+k+"k");} //Define it using super
}
class xyz{
      public static void main(String ar[])
      {d2vectors a,c;double t;d3vectors b,d; b=new d3vectors(6,8,7);c=new d2vectors(6,7);
85,149   t=c.magsqr( );System.out.println(t);   t=b.magsqr( );System.out.println(t); d = new d3vectors(1,2,
1.33 0.7 t=b.slope( );System.out.println(t);       t=b.direction( );System.out.println(t); b= new d3 vectors (6,8,7)
12,16,14 c.pt( ); b.pt( ); b.dble( ); b.pt( ); b.ttt( ); b.pt( ); c.ab(d); d.ab(c); b.ab(d);
24,32,42 } Replace k by j at all places in class d3vectors. To get same
6712  }-----------------------------------------------------------------  output use
1267   ~~Define class stack { protected String x[]=new String[10]; int sp; put take top pt }~~  super, j at
6812  import java.io.*; import java.lang.*;                                   least number of
      class stack                                                                 places
      { protected String s[]=new String[10];int sp=1;    public void put(String e) {s[sp]=e;sp++;}
        public String take( ) {sp--; return s[sp];}          public String top( ) {return s[sp-1];}
        public void pt( ){int i;for (i=1;i<sp;i++) System.out.print(s[i]+" ");}
      }
      class kapil
      {  public static void main( String args[]) throws Exception
        { String a=" ",b;int i;stack x;x=new stack( );
          DataInputStream k=new DataInputStream(System.in);
          do{a=k.readLine( );i=a.indexOf(' ');if (i!=-1) b=a.substring(0,i);else b=a;
               if (b.compareTo("put")==0)x.put(a.substring(i+1));
               if (b.compareTo("take")==0) System.out.println("The taken element "+x.take( ));
               if (b.compareTo("top")==0) System.out.println("The first element "+x.top( ));
               if (b.compareTo("print")==0) {x.pt();System.out.println( );}
          }while (b.compareTo("exit")!=0);
        }
      } put 12  put 34  put 26  put 21  (take) (top) will output 21 and 26
      class queue extends stack
      {  public void put(String e){  int i;  for (i=sp;i>1;i--) s[i]=s[i-1];          s[1]=e;sp++;  } }
```

In main use queue x; x=new queue( ); put 12 put 34 put 26 put 21 (take) (top) ⇒ 12,34

1. Define class queue extends stack{take,top} (Functions put and print are not defined)
2. Define class priority_queue extends stack{take,top}. The element taken is the biggest element. No shifting. put(12,34,26,21,23,22) (take) (top)(print) ⇒ 34,26(12,22,26,21,23)
3. Define class priority_queue extends stack{put}. (shifting permitted)
4. Define class queue extends stack{ private int front;take,top,print} (No shifting).
5. Define data type "sequence". Its operations are put(e,k), take(k), find(k), size and print. After operations put(12,1) put(37,2) put(41,1) put(67,2) put(95,3) take(4) the sequence will look as 41,67,95,37. The element taken will be 12. find(3) will return 95.
6. Define class stack extends sequence{put, take, top}. In the definition of put use super.put(e,1). In the definition of top use find(1) (no super). In take also use super.
7. Define class queue extends sequence
8. Define class queue in a normal manner using front and rear
9. Define class stack extends queue {take, top} (No shifting) 10. Class stack extends queue {put}(Shift