1. Die Methode `vorneEinfügen`:

Code in MAINKLASSE:
Wie bei `sortiertEinfügen` oder `hintenEinfügen`

Code in LISTE:

```
void vorneEinfügen (DATENELEMENT d){
   erster = new KNOTEN(erster, d);
}
```

2. Die Methode `einfügenVor`:

Code in MAINKLASSE:

Neuen Patienten einlesen und anlegen
Suchnamen des Bekannten eingeben und dessen Datenelement ermitteln
Vor diesem einfügen

Code in LISTE:

```
void einfügenVor(DATENELEMENT neu, DATENELEMENT vergleich){
   erster = erster.einfügenVor(neu, vergleich);
}
```

Code in LISTENELEMENT:

```
abstract KNOTEN einfügenVor(DATENEL. neu, DATENEL. vergleich);
```

Code in ABSCHLUSS:

```
@Override KNOTEN einfügenVor(DATENEL. neu, DATENEL. vergleich){
   return new KNOTEN(this, neu);
}
```

Code in KNOTEN:

```
@Override KNOTEN einfügenVor(DATENEL. neu, DATENEL. vergleich){
   if(daten.equals(vergleich)){
      return new KNOTEN(this, neu);
   }
   else{
      nachfolger = nachfolger.einfügenVor(neu, vergleich);
      return this;
   }
}
```

3. Die Methode `nachträglichSortieren`:

Code in MAINKLASSE:

```
else if (auswahl.equals("x")) {
   ls.nachträglichSortieren();
}
```

Code in LISTE:

```
void nachträglichSortieren(){
   if (längeGeben()>1){
      DATENELEMENT d = ersterEtfernen();
      nachträglichSortieren();
      sortiertEinfügen(d);
   }
}
```

4. Die Methode `datenVonPosGeben`:

Code in MAINKLASSE:

```
else if (auswahl.equals("x")) {
   System.out.print("Welche Position? ");
   int pos = Integer.parseInt(scanner.nextLine());
   DATENELEMENT gesucht = ls.datenVonPosGeben(pos);
   if (gesucht != null) {
      gesucht.infosAusgeben();
   }else{
      System.out.println("Nicht gefunden");
   }
}
```

Code in LISTE:

```
DATENELEMENT datenVonPosGeben(int pos) {
   if (pos < 1 || pos >= längeGeben()+1){
      return null;
   }
   return erster.datenVonPosGeben(pos);
}
```

Code in LISTENELEMENT:

```
abstract DATENELEMENT datenVonPosGeben(int pos);
```

Code in KNOTEN:

```
@Override DATENELEMENT datenVonPosGeben(int pos) {
   if (pos == 1){
      return daten;
   }else{
      return nachfolger.datenVonPosGeben(pos - 1);
   }
}
```

Code in ABSCHLUSS:

```
@Override DATENELEMENT datenVonPosGeben(int pos){
   return null;
}
```

5. Die Methode `prüfenObAlleErwachsen`:

Code in MAINKLASSE:

```
else if (auswahl.equals("x")) {
   System.out.println(ls.pOAE()? "Alle Ü18" : "auch U18");
}
```

Code in LISTE:

```
boolean prüfenObAlleErwachsen(){
   return erster.prüfenObAlleErwachsen();
}
```

Code in LISTENELEMENT:

```
abstract boolean prüfenObAlleErwachsen();
```

Code in KNOTEN:

```
@Override boolean prüfenObAlleErwachsen(){
   if (daten.getAlter()>=18){
      return nachfolger.prüfenObAlleErwachsen();
   }else{
      return false;
   }
}
```

Code in ABSCHLUSS:

```
@Override boolean prüfenObAlleErwachsen(){
   return true;
}
```

6. Die Methode `trennenNachArt`:

Code in MAINKLASSE:

```
else if (auswahl.equals("x")) {
   LISTE g = new LISTE();
   LISTE p = new LISTE();
   ls.trennenNachArt(g,p);
   System.out.println("Gesetzliche: ");
   g.infosAusgeben();
   System.out.println("Private: ");
   p.infosAusgeben();
}
```

Code in LISTE:

```
void trennenNachArt(LISTE g, LISTE p){
   erster.trennenNachArt(g,p);
}
```

Code in LISTENELEMENT:

```
abstract void trennenNachArt(LISTE g, LISTE p);
```

Code in KNOTEN:

```
@Override void trennenNachArt(LISTE g, LISTE p){
   if (daten.getGesetzlich()){
      g.sortiertEinfügen(daten);
   }else{
      p.sortiertEinfügen(daten);
   }
   nachfolger.trennenNachArt(g,p);
}
```

Code in ABSCHLUSS:

```
@Override void trennenNachArt(LISTE g, LISTE p){}
```

7. Die Methode `getHöchstesAlter`:

Code in MAINKLASSE:

```
else if (auswahl.equals("x")) {
   System.out.print("Ältester Patient ist: ");
   System.out.println(ls.getHöchstesAlter());
}
```

Code in LISTE:

```
int getHöchstesAlter(){
   return erster.getBisherigesMaxalter(0);
}
```

Code in LISTENELEMENT:

```
abstract int getBisherigesMaxalter(int bisher);
```

Code in KNOTEN:

```
@Override int getBisherigesMaxalter(int bisher){
   bisher = Math.max(bisher, daten.getAlter());
   return nachfolger.getBisherigesMaxalter(bisher);
}
```

Code in ABSCHLUSS:

```
@Override int getBisherigesMaxalter(int bisher){
   return bisher;
}
```

8. Die Methode `umdrehen`:

Code in MAINKLASSE:

```
else if (auswahl.equals("x")) {
   ls.infosAusgeben();
   ls.umdrehen();
   System.out.println("Umgedreht: ");
   ls.infosAusgeben();
}
```

Code in LISTE:

```
void umdrehen(){
   erster = erster.umdrehen(new ABSCHLUSS());
}
```

Code in LISTENELEMENT:

```
abstract LISTENELEMENT umdrehen(LISTENELEMENT l);
```

Code in KNOTEN:

```
@Override LISTENELEMENT umdrehen(LISTENELEMENT l){
   return nachfolger.umdrehen(new KNOTEN(l, daten));
}
```

Code in ABSCHLUSS:

```
@Override LISTENELEMENT umdrehen(LISTENELEMENT l){
   return l;
}
```