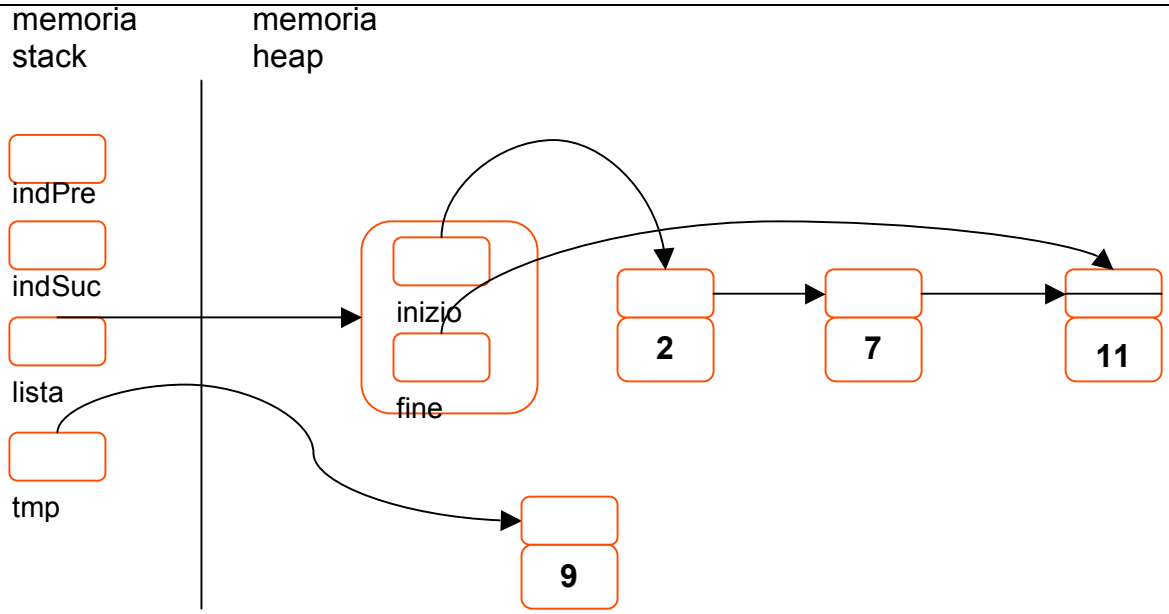
Corso di laurea in Informatica AA 2006/07 Programmazione, M.A. Alberti

II Appello 2006/07 - 19 febbraio 2007

Cognome **Nome**

Matricola

1



Osservate la situazione dello stack delle chiamate dei metodi e dello heap schematizzato dalla figura, in cui si fa riferimento alla classe `ListaOrdinata` con la classe interna `NodoLista`:

```
public class ListaOrdinata {
    private NodoLista inizio, fine;
    private static class NodoLista {
       Comparable dato;
       NodoLista pros;
    }    …… }
```

La variabile `lista` sullo stack è il riferimento a una lista raffigurata dopo un certo numero di inserimenti. Si noti che la lista è mantenuta in ordine crescente. Si noti anche che la figura semplifica la struttura indicando il valore e non l'oggetto di tipo `Comperable` (cioè nel caso specifico indicando i valori `int` anziché i riferimenti a oggetti `Integer`).

1. Dichiarate e inizializzate due variabili `indPrec` e `indSuc` che verranno usate come indice di scorrimento della lista per trovare la posizione giusta in cui inserire un nuovo nodo (ad esempio contenente il valore 9). Si noti che come il nome suggerisce `indPrec` precede sempre `indSuc`:

```
NodoLista indPrec = null, indSuc = lista.inizio;
```

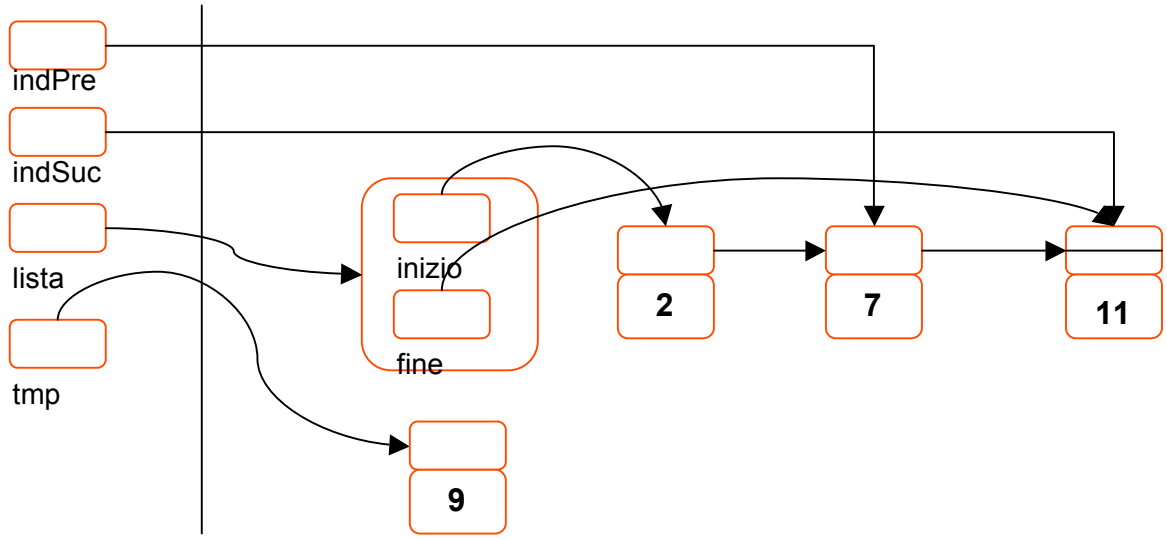
2. Scrivete il ciclo che controlli in che posizione occorre inserire un nuovo nodo nella lista ordinata

```
while (indSuc != null && indSuc.dato.compareTo(tmp.dato) < 0) {
          indPrec = indSuc;
          indSuc = indSuc.pros;
}
```

3. Inserite il nuovo nodo puntato da tmp nella posizione corretta:

```
tmp.pros = indSuc;
if (indPrec == null)  //inserimento all'inizio della lista
      inizio = tmp;
else                  //inserimento dopo il nodo riferito da indPrec
      indPrec.pros = tmp;
```

4. Disegnate sulla figura lo stato delle variabili sullo stack rispetto alla struttura al momento in cui avendo avanzato gli indici di scorrimento siete in grado di decidere dove inserire il nuovo nodo.



2

Implementare il metodo `public int trova(Comparable x)` che restituisce la posizione dell'oggetto passato come parametro se questo è presente nella lista o `0` nel caso non lo sia.

```
public int trova(Comparable x)
{
   NodoLista p = inizio;
   int posizione = 1;
   while (p != null && p.dato.compareTo(x)<0)
   {
      p = p.pros;
      posizione++;
   }
   if (p==null || p.dato.compareTo(x)>0) //se non c'e' il nodo con il dato
      return 0;
   else
      return posizione;
}
```

3

Data la classe `Dipendente` con i costruttori elencati:

```
class Dipendente {
   String nome;
   Dipendente(){
      nome = new String();
   }
   Dipendente(String unNome) {
      nome = unNome;
   }
}
```

1. E' possibile cambiare il valore del membro `nome` di un oggetto già creato? **SI** NO
2. In caso affermativo scrivere le istruzioni necessarie, in caso negativo spiegare:

```
public void modificaNome (String nuovoNome)
{
   nome = nuovoNome;
}
```

3. Per estendere la classe con una specializzazione, introduciamo la sottoclasse `Impiegato`, completare quindi il codice seguente opportunamente:

```
class Impiegato extends Dipendente {
   private int settore;
   // 1, 2, 3 ad esempio
   Impiegato()
   {
      super();
      settore = 1; // default
   }
```

```
Impiegato (String unNome, int unSettore)
{
   super(unNome);
   settore = unSettore;
}


}
```

Supponendo di aver eseguito l'istruzione: `Impiegato imp = new Impiegato();`

4. E' possibile accedere al campo `nome` della superclasse? **SI** NO
5. Per accedere al campo `nome` è corretta l'espressione: `imp.nome`? SI **NO** Esistono altri modi? **SI** NO
6. Definire il codice di possibili eventuali alternative, per accedere al campo. **Si puo' accedere al campo nome solo in modo indiretto perche' il campo non viene ereditato. Ad esempio definendo un metodo pubblico come:**

```
public String riportaNome () {
   return nome;
}
```

7. In generale, in una sottoclasse è possibile accedere direttamente ad un campo privato di un'istanza della superclasse? SI **NO**
8. In una sottoclasse è possibile usare un metodo privato della superclasse? SI **NO**

4

Scrivere il metodo `init` che inizializza con numeri dispari un array di interi di dimensione `MAX`, passato come parametro. L'array inizializzato andrà riportato all'ambiente chiamante. A scelta si usi un ciclo `for` o un ciclo `while`:

```
public static int[] init(int[] a)
{
   for (int i=0; i < a.length; i++)
      a[i]=2*i + 1;
   return a;
}
```

oppure con un ciclo while:

```
public static int[] init(int[] a)
{
   int i=0;
   while (i < MAX)
      a[i]=2*++i + 1;
   return a;
}
```

5

Implementare l'interfaccia `Enumeration` del pacakge `java.util` per la classe `EnumerationDemo`:

```
public class EnumerationDemo implements java.util.Enumeration {
   private String[] mucchio =
      {"palla", "carta", "corda", "sasso", "fiore"};
   private int indice = 0;
   public boolean hasMoreElements()
   {
   if (indice == mazzo.length)
      return false;
   else
      return true;
   }

   public Object nextElement() {
      return (Object)mazzo[indice++];
   }
```

6

Data la funzione ricorsiva seguente:

```
public static int f(int m, int n) {
   if (m > 3)
      return n;
   else if (m > n)
      return 1 + f(m+2, n-1);
   else
      return 1 - f(n+1, m-2);
}
```

calcolare il valore riportato dalla chiamata `f(1, 2)`: **2**

calcolare il valore riportato dalla chiamata `f(1, -3)`: **-3**

calcolare il valore riportato dalla chiamata `f(2, 4)`: **1**

7

Utilizzando la classe `StringTokenizer` completare il codice seguente e implementare un ciclo for necessario per inizializzare l'array di stringhe `frase`. Ad esempio: se in input è data la frase "`questo è un esame`" allora l'array `frase` dovrà essere inizializzato a `[questo, è, un, esame]`.

```
String riga = in.readLine("input una riga di testo: ");
StringTokenizer processore_di_riga = new StringTokenizer(riga);

String [] frase = new String[processore_di_riga.countTokens()];
for (int i = 0; processore_di_riga.hasMoreTokens(); i++)
      frase[i] = processore_di_riga.nextToken();
```

8

Scrivere un metodo ricorsivo `funzione` che calcoli l'n-esimo numero della seguente sequenza:

```
1     3     7     13    21    31    43    57    73    91    111          valori
0     1     2     3     4     5    posizioni
```

```
public static int funzione(int n)
{
   if (n == 0)
      return 1;
   else
      return 2*n + funzione(n - 1);
}
```

Se si numerano le posizioni da 1 allora la soluzione al problema diventa:

```
1     3     7     13    21    31    43    57    73    91    111          valori
1     2     3     4     5     6    posizioni
```

```
public static int funzione(int n)
{
   if (n == 1)
      return 1;
   else
      return 2*(n-1) + funzione(n - 1);
}
```

9

Scrivere il metodo `alfabetico?` che riporta un valore booleano per indicare se il parametro d'ingresso è un carattere alfabetico compreso tra a-z o tra A-Z (minuscolo o maiuscolo)

```
public boolean alfabetico? (char ch)
{
  return ( (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') );
}
```