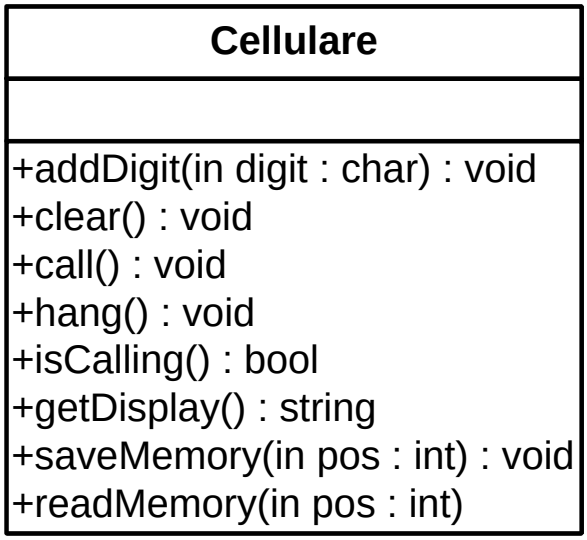
Esempi di esercizi d'esame

Cellulare – Testo esercizio: 1.Specifiche

- Si richiede di realizzare una classe che implementa il funzionamento di un telefono cellulare
- Tale classe dovrà consentire di:
  - Comporre un numero telefonico, una cifra alla volta (verificando che le cifre siano valide)
  - Leggere il valore del display (stringa con il numero corrente)
  - Cancellare il numero corrente
  - Effettuare una chiamata
  - Terminare una chiamata
  - Dire se il telefono ha una chiamata in corso
  - Memorizzare il numero corrente in una cella specificata (compresa fra 1 e 10)
  - Leggere come numero corrente il numero memorizzato in una cella specificata (fra 1 e 10)

Cellulare – Testo esercizio: 2.Diagramma UML

- **Il diagramma UML della classe è riportato qui sotto (la parola chiave in davanti al nome dei parametri dice semplicemente che sono parametri passati per valore, lo standard per Java)**



Cellulare – Testo esercizio: 3.Classe di test

```
public class EsempioCellulare
{
  public static void main(String args[])
  {
    Cellulare cell = new Cellulare()
    cell.addDigit('3');
    cell.addDigit('4');
    cell.addDigit('7');
    System.out.println(cell.getDisplay());
    cell.saveMemory(5);
    cell.clear()
    cell.addDigit('6');
    cell.addDigit('6');
    cell.addDigit('1');
    cell.readMemory(5);
    System.out.println(cell.getDisplay());
  }
}
```

Cellulare – Soluzione: 1.Scelte implementative

- Dobbiamo definire la composizione dello stato
- Abbiamo bisogno di memorizzare sostanzialmente due cose:
  - Il numero corrente
  - Le dieci celle di memoria per i numeri
- La cosa più comoda è memorizzare i numeri come stringhe: è più facile gestire l'aggiunta delle cifre e non dobbiamo fare conversioni per restituire il display
- Quindi useremo due attributi: num di tipo String e, mem, array 0..9 di String
- Dovremo ricordarci di creare l'array e le stringhe nel costruttore
- Abbiamo poi bisogno di un boolean per dire se c'è una chiamata in corso o no

Cellulare – Soluzione: 2.Implementazione/1

```
public class Cellulare
{
  private String num;
  private String[] mem;
  private boolean calling;
  public Cellulare()
  {
   calling = false;
   num = "";
   mem = new String[10];
   for (int i=0; i<10; i++)
    mem[i] = "";
  }
  public void addDigit(char ch)
  {
    if ((ch>='0') & (ch<='9')) num = num + ch;
  }
…
```

Cellulare – Soluzione: 2.Implementazione/2

```
…
   public void clear()
   {
     num = “”;
   }
   public void call()
   {
     calling = true;
   }
   public void hang()
   {
     calling = false;
   }
   public boolean isCalling()
   {
     return calling;
   }
…
```


Cellulare – Soluzione: 2.Implementazione/3

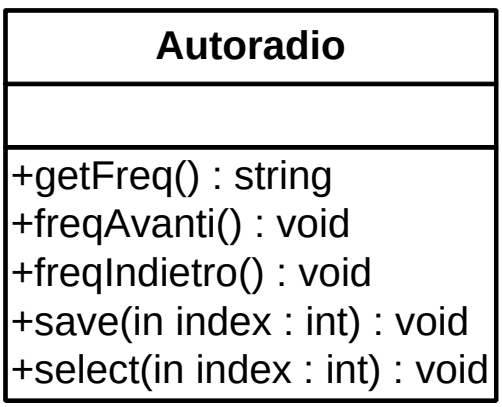
```
…
   public void getDisplay()
   {
     return num;
   }
   public void saveMemory(int pos)
   {
     mem[pos-1] = num;
   }
   public void readMemory(int pos)
   {
     num = mem[pos-1];
   }
}
```

Autoradio - Testo esercizio: 1.Specifiche

- **Si richiede di realizzare una classe che implementa il funzionamento di un'autoradio FM in grado di sintonizzarsi su una frequenza compresa fra 87.5 e 108.00 MhZ a passi di 0,5 MHz, dotata di 6 preselezioni.**
- **Tale classe dovrà consentire di:**
  - **Far avanzare la frequenza corrente**
  - **Far indietreggiare la frequenza corrente**
  - **Restituire la frequenza corrente come stringa**
  - **Salvare la frequenza corrente in un adelle 6 preselezioni**
  - **Leggere la frequenza corrente da una delle 6 preselezioni**

Autoradio – Testo esercizio: 2.Diagramma UML

- **Il diagramma UML della classe è riportato qui sotto (la parola chiave in davanti al nome dei parametri dice semplicemente che sono parametri passati per valore, lo standard per Java)**



Autoradio – Testo esercizio: 3.Classe di test

```
public class EsempioAutoradio
{
  public static void main(String args[])
  {
    Autoradio auto = new Autoradio()
    for (int i>0; i<10; i++)
      auto.freqAvanti();
    auto.save(1);
    auto.freqIndietro();
    System.out.println("Frequenza: "+cell.getFreq());
    auto.save(2);
    auto.freqAvanti();
    auto.select(1);
    System.out.println("Frequenza: "+cell.getFreq());
    auto.select(2);
    System.out.println("Frequenza: "+cell.getFreq());
  }
}
```


Autoradio – Soluzione: 1.Scelte implementative

- Dobbiamo definire la composizione dello stato
- Abbiamo bisogno di memorizzare sostanzialmente due cose:
  - La frequenza corrente
  - Le sei celle di memoria per le frequenze
- Una possibilità, non l'unica, è memorizzare le frequenze come interi moltiplicandole per 10
- Quindi useremo due attributi: freq di tipo int e sel, array 0..5 di int
- Dovremo ricordarci di creare l'array nel costruttore

Autoradio – Soluzione: 2.Implementazione/1

```
public class Autoradio
{
  private int freq;
  private int sel[];
  public Autoradio()
  {
    freq = 875;
    sel = new int[6];
    for (int i=0; i<6; i++)
      sel[i]=freq;
  }
  public String getFreq()
  {
    return ""+(freq / 10)+"."+(freq % 10);
  }
```


Autoradio – Soluzione: 2.Implementazione/2

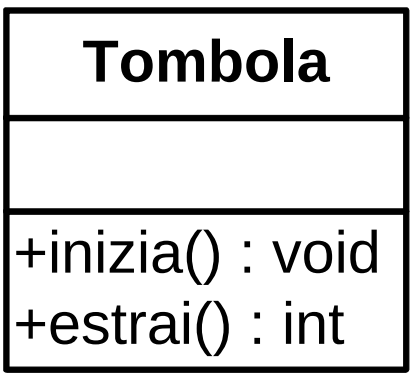
```
  public void freqMeno()
  {
    if (freq>875)
      freq=freq-5;
  }
  public void freqPiu()
  {
    if (freq<1080)
      freq=freq+5;
  }
  public void save(int index)
  {
    if ((index>=0) & (index<=5)) sel[index-1]=freq;
  }
  public void select(int index)
  {
    if ((index>=0) & (index<=5)) freq=sel[index-1];
  }
}
```

Tombola – Testo esercizio: 1.Specifiche

- **Si richiede di realizzare una classe che implementa il gioco della tombola**
- **Tale classe dovrà consentire di:**
  - **Dichiarare l'inizio di una partita**
  - **Estrarre più volte (fino ad un massimo di 90) in modo casuale un numero compreso fra 1 e 90, garantendo che in una partita non venga estratto per due volte lo stesso numero**
  - **Se sono già state fatte 90 estrazioni durante la partita viene restituito il numero 0**

Tombola – Testo esercizio: 2.Diagramma UML

- **Il diagramma UML della classe è riportato qui sotto (la parola chiave in davanti al nome dei parametri dice semplicemente che sono parametri passati per valore, lo standard per Java)**



Tombola – Testo esercizio: 3.Classe di test

```
public class EsempioTombola
{
  public static void main(String args[])
  {
    Tombola t = new Tombola();
    t.inizia();
    for (int i=0; i<90; i++)
      System.out.println("Estratto: "
        + t.estrai());
  }
}
```


Tombola – Soluzione: 1.Scelte implementative

- Dobbiamo definire la composizione dello stato
- Abbiamo bisogno di memorizzare sostanzialmente due cose:
  - Quante estrazioni sono state fatte dall'inizio della partita
  - Quali sono i numeri già estratti
- Per memorizzare i numeri estratti possiamo usare una array di 90 boolean
- Per memorizzare il numero di estrazioni fatte è sufficiente un intero

Tombola – Soluzione: 2.Implementazione/1

```
public class Tombola
{
  private boolean estratti[];
  private int numEstrazioni;
  public Tombola()
  {
    estratti = new boolean[90];
    inizia();
  }
  public void inizia ()
  {
    for (int i=0; i<90; i++)
      estratti[i] = false;
    numEstratti = 0;
  }
  …
```


Tombola – Soluzione: 2.Implementazione/2

```
  …
  public int estrai()
  {
    boolean ok = false; int n = -1;
     if (numEstratti<90)
      while (!ok)
      {
        n = (int)Math.round(Math.random()*89);
        if (estratti[n]==false)
        {
          ok = true;
          estratti[n] = true;
          numEstratti++;
        }
      }
    return n+1;
  }
}
```