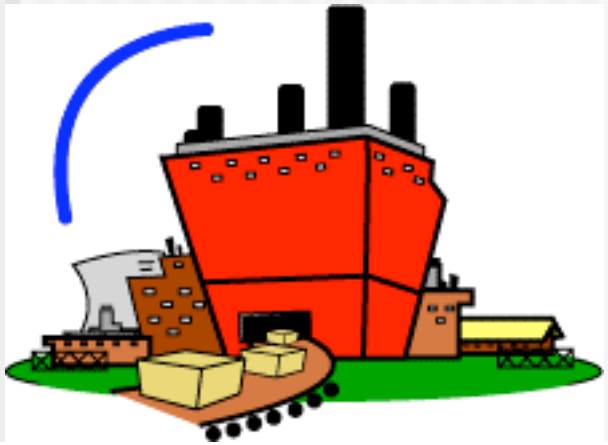
Computer Science 101

Lecture 16

Classes 2



Contents

Symbolic constants

Practice defining a class - Time24Hr class

`toString()` method

`public/private` modifier.

accessor methods

mutator methods

Coursebook: pages **§14.1 - §14.4.3, §14.4.5**

Declaring Symbolic Constants

Format:

<accessability> static final <type> <id> = <value>;

Ex:

private static final int MAX_NODES = 20;

public static final double THRESHOLD = 35.3;

Style Rule for Naming Constants

All the letters in the constant's id are upper case. If there is more than one word in the id then they are separated by underscores ("_").

Ex.: ***MAX_NODES,*** and ***THRESHOLD.***

Use of Symbolic Constants

Ex:

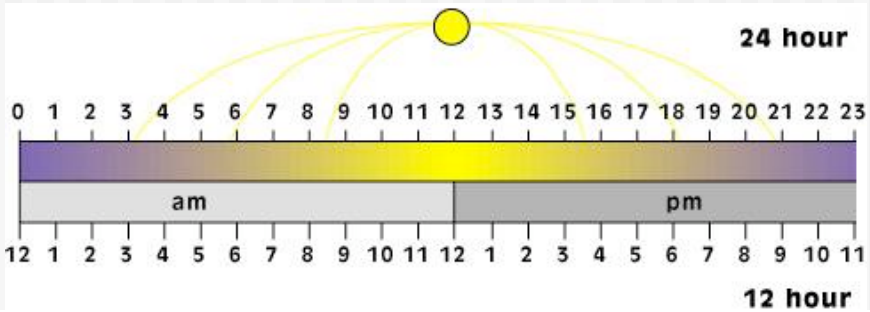
```
private static final int MAX_COUNT = 20;
int [ ] ages = new int [MAX_COUNT];
for (int i = 0; i < MAX_COUNT; i++)  {
        …
}
```

`Time24Hr` - instance variables

Consider defining a class to represent hours and minutes on a 24 hour clock.

```
public class Time24Hr {
    private int hour;
    private int minutes;
}
```

The instance variables represent the information which all instances of the `Time24Hr` class will contain.



`Time24Hr` - constructor

The constructor will be called whenever a new instance of the **`Time24Hr`** class is required.

```
public class Time24Hr {
   private int hour;
   private int minutes;

   public Time24Hr(int h, int m) {
     hour = h;
     minutes = m;
   }

}
```

`Time24Hr` - instance methods

Decide which operations are possible on objects of this class?

`setHour()`: change the value stored in the hour variable of the **`Time24Hr`** object.

`setMinutes()`: change the value stored in the minutes variable of the **`Time24Hr`** object.

`isAM()`: return true if the time is an AM time, return false otherwise.

`addToMinutes()`: add some minutes to the time stored in the object.

Time24Hr - instance methods

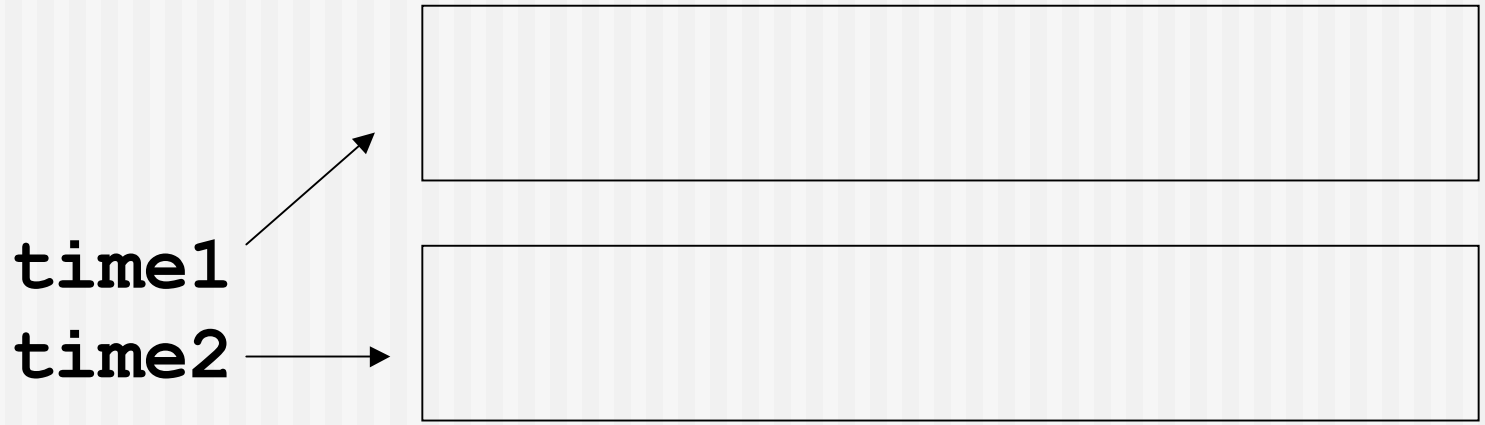
```
public class Time24Hr {
   private int hour;
   private int minutes;
   public Time24Hr(int h, int m) {
      hour = h;
      minutes = m;
   }
   public void setHour(int h) {
      hour = h;
   }
   public void setMinutes(int m) {
      minutes = m;
   }
```

Time24Hr - instance methods

```
    public boolean isAM() {
      if (hour<12) {
           return true;
      }
      return false;
    }

   public void addToMinutes(int moreM) {
      minutes = minutes + moreM;
      hour = hour + minutes/60;
      hour = hour%24;
      minutes = minutes%60;
   }
}
```

Ex01 - Draw the two objects

```
public class ProgramClassL15 {
   public void start() {
       int hrs = 10;
       int mins = 15;
       Time24Hr time1 = new Time24Hr(hrs, mins);
       Time24Hr time2 = new Time24Hr(9, 20);
   }
}
```



Ex02 - What is the output?

```
public class ProgramClassL15 {
 public void start() {
      int hrs = 10;
      int mins = 15;
      Time24Hr time1 = new Time24Hr(hrs, mins);
      Time24Hr time2 = new Time24Hr(9, 20);
      time1.setMinutes(45);
      time2.setHour(23);
      time2.addToMinutes(360);
      if (time1.isAM()) {
        System.out.println("time1 - morning");
      }
      if (time2.isAM()) {
        System.out.println("time2 - morning");
      }
}}
```

`toString()` method

In a class definition the **`toString()`** instance method returns a string representation of an object. This is convenient for displaying the value of an object. The general format of the **`toString()`** method is:

```
public String toString() {
    String desc = "";
    desc = desc + ... ;
    ...
    desc = desc + ... ;
    return desc;
}
```

Example 1 - toString()

```
public class Chocolate {
   private int code;
   private String description;
   private double price;
   public Chocolate(int c, String desc, double p){
      code = c;
      description = desc;
      price = p;
   }
   public double getPrice() {
       return price;
   }
   public double getPrice20() {
       return price * 20 * 0.8;
   }
```

Example 1 - toString()

```
    public void setDescription(String desc) {
        description = desc;
    }

    public String toString() {
        String chocS = "";
        chocS = chocS + "Code: " + code + ": ";
        chocS = chocS + description;
        chocS = chocS + ", $" + price;
        return chocS ;
    }
}
```

Example 1 - displaying the object information


```
public class ProgramClassL15 {
 public void start() {
   Chocolate choc4;
   String chocDesc;
   choc4 = new Chocolate(113, "Nuts", 1.50);
   chocDesc = choc4.toString();
   System.out.println(chocDesc);

   choc4.setDescription("Lots of Nuts");
   System.out.println(choc4.toString());
 }
}
```

Example 2 - toString()

```
public class Time24Hr {
   private int hour;
   private int minutes;
   public Time24Hr(int h, int m) {
      hour = h;
      minutes = m;
   }
   public void setHour(int h) {
      hour = h;
   }
   public boolean isAM() {
      if (hour<12) {
      }     return true;
      return false;
   }
```

Example 2 - toString()

```
  public void setMinutes(int m) {
     minutes = m;
  }
  public void addToMinutes(int moreM) {
     minutes = minutes + moreM;
     hour = hour + minutes/60;
     hour = hour%24;
     minutes = minutes%60;
  }
  public String toString() {
      String timeS = "";
      timeS = timeS + "Hour: " + hour;
      timeS = timeS + ", Minutes: " + minutes;
      return timeS ;
  }
}
```

Ex03 - What is the output?

```
public class ProgramClassL15 {
  public void start() {
      Time24Hr time1;
      time1 = new Time24Hr(9, 20);
      System.out.println("1: " + time1.toString());

      time1.addToMinutes(65);
      System.out.println("2: " + time1.toString());

  }
}
```

accessability modifiers

`private:` the instance method or instance variable can only be accessed inside the class in which it is defined.

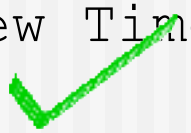
`public:` the instance method or instance variable can be accessed inside the class and outside the class in which it is defined e.g. it can be accessed from other classes.

Why is the **`start()`** instance method in the class **`ProgramClassL15`** declared as **`public`**?

`public` instance methods

```
public class Time24Hr {
   private int hour;
   private int minutes;
   public Time24Hr(int h, int m) {
      hour = h;
      minutes = m;
   }
   public void setHour(int h) {
      hour = h;
   }
      //Rest of class not shown
}
```

```
public class ProgramClassL15 {
  public void start() {
      Time24Hr time1 = new Time24Hr(9, 20);
      time1.setHour(8);
  }
}
```



private methods

```
public class Time24Hr {
   private int hour;
   private int minutes;
   public Time24Hr(int h, int m) {
      hour = h;
      minutes = m;
   }
   private void setHour(int h) {
      hour = h;
   }
      //Rest of class not shown here
}
```

```
public class ProgramClassL15 {
  public void start() {
      Time24Hr time1 = new Time24Hr(9, 20);
      time1.setHour(8);
  }
}
```

`private` instance variables

In CS 101 class definitions we ALWAYS declare ALL instance variables as **`private`**.

```
public class Time24Hr {
    private int hour;
    private int minutes;
    public Time24Hr(int h, int m) {
        hour = h;
        minutes = m;
    }
        //Rest of class not shown
}
```

```
public class ProgramClassL15 {
   public void start() {
        Time24Hr time1 = new Time24Hr(9, 20);
        time1.hour = 11;
   }
}
```

Accessor (get) methods

To obtain values from the **`private`** instance variables, `public` accessor methods (**`get()`** methods) are defined:

```
public class Time24Hr {
   private int hour;
   private int minutes;
   public Time24Hr(int h, int m) {
      hour = h;
      minutes = m;
   }

   public int getHour() {
     return hour;
   }
   public int getMinutes() {
      return minutes;
   }
            //Rest of class is not shown here
}
```

Mutator (set) methods

To assign values to the **`private`** instance variables, **`public`** mutator methods (**`set()`** methods) are defined.

```
public class Time24Hr {
   private int hour;
   private int minutes;
   public Time24Hr(int h, int m) {
      hour = h;
      minutes = m;
   }
   public void setHour(int h) {
      hour = h;
   }
   public void setMinutes(int m) {
     minutes = m;
   }
            //Rest of class is not shown here
}
```

Ex04-private instance variables

By looking at the following code think of a reason why all instance variables are declared as private.

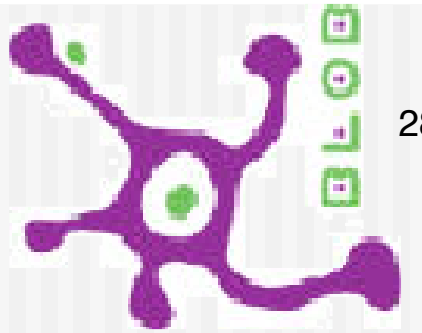
```
public class Time24Hr {
   private int hour;
   private int minutes;

   public void setHour(int hr) {
        if (hr>23) {
           hr = hr%24;
        }
        if (hr<0) {
           hr = 0;
        }
        hour = hr;
   }
              //Rest of class is not shown here
}
```

Ex05 -get and set methods

Define the **`get()`** and **`set()`** methods for the instance variable, **`uPI`**:

```
public class MarkAss1 {
   private int mark;
   private String uPI;
   public String getUPI() {
         return uPI;
   }
   public void setUPI(String aUPI) {
         uPI = aUPI ;
   }
 }
```

If I want the **`uPI`** instance variable to be read-only what can I do? _____



Ex06 - classes

Study the **skeleton** of the **`Blob`** class definition:

```
public class Blob {
 private String theText;                          //1

 private int value;                               //2

 public Blob(String s, int num) {… }              //3

 public int getValue() {… }                       //4

 public void addToText(String s) {… }             //5

 public boolean isPositive() {… }                 //6
}
```

Ex06 - classes

Identify:

1. A constructor method: ____
2. Instance variable: ____
3. Instance method returning a boolean: ____
4. Instance method with a String parameter: ___
5. Instance method which returns a int: ____
6. A correct statement to create a Blob object?
   a) Blob blob1 = new Blob("Hi");
   b) Blob blob1 = new Blob("Hi", 3456);
   c) Blob blob1 = new Blob("Hi", true); ____
7. A correct call to an instance method
   a) String s = blob1.addToText("Hi");
   b) System.out.println(blob1.equals(blob2));
   c) String s2 = blob1.getValue(); ____

Ex07 - Constructor problem

Is there a problem with the following constructor?

```
public class MarkAss1 {
   private int mark;
   private String uPI;

   public MarkAss1(String aUPI, int aMark) {
       aMark = mark;
       aUPI = uPI;
   }
}
```

Ex08 - `toString()` problem

Is there a problem with the following `toString()` method?

```
public class MarkAss1 {
   private int mark;
   private String uPI;
   public MarkAss1(String aUPI, int aMark) {
       mark = aMark;
       uPI = aUPI;
   }
   public void toString() {
       String markDesc = "";
       markDesc = markDesc + uPI + ": mark";
       markDesc = markDesc + mark;
       return markDesc;
   }
}
```

What you need to know

- Structure of a class definition
  - Instance variables
  - Constructors
  - Instance methods
- `toString()` method
- `private` instance variables
- Define constructors
- Define accessor methods
- Define mutator methods