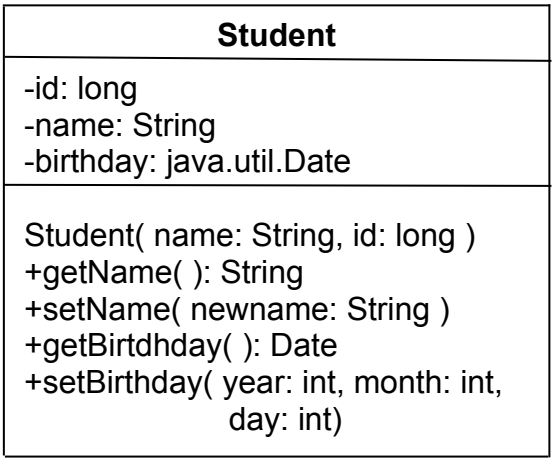
A Student has a name, id, and birthday as shown in this *UML Class Diagram*.



1. Complete the class declaration, and declare the attributes of a Student

```
import _____________________;
_____________ _____________ Student {
    _____________ _____________________________ ;
    _____________ _____________________________ ;
    _____________ _____________________________ ;
```

2. Write an *accessor* method for the Student's name attribute.

```
public _____________ _____________( ) {


}
```

3. Write an *mutator* method for the Student's name attribute. Do not accept a new name (the parameter) if it is null or an empty string. In that case, do nothing (better solution is to throw an Exception, which we haven't studied yet.)

```
public _____________ _____________ ( String _____________ ) {



}
```

4. Write a *constructor* so that we can initialize students like this:

```
Long id = 5810123456L;
Student pee = new Student("Pirawat", id);
```

5. Write an `equals` method that returns true of two students have the same **`id`**. Complete the Javadoc.

Use the **4-step pattern for equals**: 1) test if parameter is null, 2) test parameter reference is same type as this object (`x.getClass()`, 3) *cast* the parameter to a reference of type Student, 4) compare attributes as required. Also complete the Javadoc.

```
/**
 * Test if two students are equal.
 * _____________ obj is another object to compare to this one
 * _____________ true iff obj is a Student with same id as this Student
 */
public boolean equals( Object  obj ) {

     ____________________________________________________________;

     ____________________________________________________________;

     ____________________________________________________________;

     ____________________________________________________________;

     ____________________________________________________________;
}
```

6. Correct this **`toString`** method. Write your changes in the code.

```
/** Return a string representation, such as "Cat [5610541234]" */
public void toString( ) {
     System.out.println( this.name + " [" + this.id + "]");


}
```

7. Our Student class has a birthday attribute with "get" and "set" methods:

```
public class Student {
    private Date birthday;
    /** Get the student's birthday. */
    public Date getBirthday( ) {
        return this.birthday;
    }
    /** Set the student's birthday. */
    public void setBirthday(int year, int month, int day) {
        // Date constructor is weird.
        // year value is year-1900, e.g. Year 2000 is 100
        // month value is 0=January, 1=February, ..., 12=Decemeber
        this.birthday = new Date(year-1900, month-1, day);
    }
```

However, Java **strongly discourages** the use of the Date class and encourages using LocalDate instead.

We cannot change the *method signatures* for getBirthday() and setBirthday() because other classes are using those methods! *"Method signature"* means how the method appears: its name, parameters, visibility, and return type.

Fortunately, our code ***encapsulates*** *and* ***hides*** the birthday attribute.

Therefore, we can *change the internal implementation* as long as we don't change the method signatures.

Modify the code so that the Student's birthday is a java.time.LocalDate instead of Date.

a) change the **`birthday`** attribute from Date to LocalDate. Since birthday is private, other classes are not affected by this change.

b) modify **`getBirtthday()`** to create a new Date object and return it. **`LocalDate`** has these methods:

getYear() returns the year

getMonthValue() returns the month as an integer. 1 = January, 12 = December

getDayOfMonth( ) returns the day of the month

Use these methods to create a new Date object and return it.

c) modify **`setBirthday`**() so that it updates birthday as a LocalDate object instead of Date.

LocalDate has a *factory method* to create a new object: LocalDate.of(year, month, day)

Write your solution in BlueJ first so you get the syntax correct; then write it here:

```
import java.util.Date;
import java.time.LocalDate;
public class Student {
    private ___________ birthday;
    /** Get the student's birthday. */
    public Date getBirthday( ) {
        ____________________________________________________________
        ____________________________________________________________
    }
    /** Set the student's birthday.
     * @param year is the year of birth, e.g. 2001.
     * @param month is the birth month, 1=January, .., 12=December.
     * @param day is the day of the month.
     */
    public void setBirthday(int year, int month, int day) {
        ____________________________________________________________
        ____________________________________________________________
    }
```

8. For each item in the left column, identify the kind of thing is represents using items in the right column.

`char`	attribute of an *object*
`Character`	class
`List`	instance method
`System`	interface
`java.lang`	package
`java.lang.System`	primitive type
`length() in "Harry".length()`	static attribute (attribute of a *class*)
`org.junit`	static constant
`java.lang.Math.PI`	static (class) method
`java.lang.Math.sqrt( )`	variable name
`System.in`	
`System.in.read()`	
`LocalDate.now( )`	
`next() in Scanner class`	

9. Which *package* contains each of these classes?

The choices are `java.io`, `java.lang`, `java.time`, and `java.util`.

_____ `InputStream`

_____ `File`

_____ `PrintStream`

_____ `Math`

_____ `Double`

_____ `String`

_____ `Collections` – *a class that provides static methods for working with collections*

_____ `ArrayList`

_____ `Date`

_____ `LocalDate`

_____ *`Runnable`* – *an interface with a single method named* `void run( )`

_____ `System` – *provides access to operating system resources & services*