COMP 1409 – Summary of Session 3 Concepts

- Parts of a class. In session 2 we looked at instance variables (fields), class constructors, parameters, the assignment operator and accessor methods. In session 3 we continued our investigation of the parts of a class by examining mutator methods and data validation.
- Mutator methods. These are methods that change the state of an object by changing the value in one or more of the instance variables. They typically receive parameters and contain assignment statements. Mutator methods that replace the contents of one of the instance variables start with the word "set" with the name of the method corresponding with the name of the relevant field.
- Methods that display on the screen. To display something on the screen (terminal window in BlueJ) we pass whatever we need to display to System.out.println(), e.g. `System.out.println("This is a string being displayed.");`
- String concatenation. Concatenation is the joining of two strings to create a third string. The **+** symbol does two things in Java. It adds two numbers together, e.g. `4 + 5` results in `9`. It concatenates strings, e.g. `"hello" + "there"` results in `"hellothere"`. Other data types can be joined to strings for output.
- The operation proceeds from left to right, except that an expression inside parentheses will be processed first.

  `System.out.println("hello" + 4 + 5)` displays `hello45`.

  `System.out.println(4 + 5 + "hello")` displays `9hello`.

  `System.out.println("hello"+(4 + 5))` displays `hello9`.

  `System.out.println(("hello" + 4) + (5 + 5))` displays `hello410`

  `System.out.println(5 + 5)` displays `10`

- Relational operators. These are the comparison operators used with numeric data, e.g. equals `( == )`, not equals `( != )`, greater than `( > )`, less than `( < )`, less than or equal to `( <= )`, greater than or equal to `( >= )`. They take two operands, and result in a boolean value, `true` or `false`.
- Making choices. The `if` statement is used to make a decision and cause the execution of the program to branch.

```
if (boolean condition) {
    // statements to execute if the condition is true
}
else { // else is optional
    // statements to execute if the condition is false
}
```

- Data validation. We need to ensure that objects are created in valid initial states. An if-else construction is typically used in a class constructor and/or a mutator method.

```
/**
  Sets the price of a ticket
  @param newPrice ticket price to set
*/
public void setPrice(int newPrice)
{
   if(newPrice >= 0) { // can't be negative
      price = newPrice;
   }
   else {
      System.out.println("Invalid price!");
   }
}
```