Assignment 1

Common mistakes and lessons to learn

Java programming

Class Inheritance

Data Structures, Fall 2018

TA: Marija Stanojevic

Common mistakes - technical

1. No README.txt file: Purpose of this file is to **tell someone who doesn't know your code what is it about and how to run it**.
2. No comments: Purpose of comments is to clarify your code and assumptions you've made. Write comments before **class/function/complicated line of code/whenever you made some assumption**. Examples:
    a. //computePay computes salary per month
    b. //This class contain information about Person
    c. //Field hoursWorked contain information about hours worked within a week
3. Don't forget comments on top of each file according to instructions.
4. One class per file; file name should be the same as class name

Common mistakes - code

1. Class/Constructor naming: **ClassNameRule** - each word starts with capital
2. Variable/argument/function/field naming: **variableNameRule** - first word starts with lowercase, other words starts with capital. Examples:
   a. ssn, getSsn(), gpa, setGpa(), telephoneNumber()
3. Package naming: **packagenamerule** - all letters lowercase
4. Don’t use _ anywhere in java names
5. Don’t use parentheses with return or when not needed. Examples:
   a. ~~return (name);~~ => return name;
   b. ~~String toPrint = (“Name is ” + name) + (“.”);~~ => String toPrint = “Name is ” + name + “.”;
   c. ~~a = b + (c * d);~~ => a = b + c * d;
6. Check operators priorities if you are not sure where parentheses should be.

Common mistakes - code (2)

1. **Getters/setters naming**: getFieldName(); setFieldName();
2. **Code formatting:**
    a. NetBeans shortcut: Alt + Shift + F
    b. Eclipse shortcut: Ctrl + Shift + F
    c. IntelliJIDE shortcut: Ctrl + Alt + L
    d. Important rules:
        i. { should go on the end of previous line and not as standalone in new line
        ii. There should be space between operators and operands
        iii. When you open { bracket, next lines of code need to be indented until you close that block with }. For indentation you can use TAB or 4 spaces.
        iv. Between two functions leave on line empty
        v. Don’t leave lines empty inside of the function, except when you do multiple things in the function and you want to separate them.

Common mistakes - code (3)

1. All fields need to be declared private (**private String name;**)
2. To access field from other classes use getters/setters
3. You don’t need to use **this** with **fieldName** or **funcName** if you are in the class where those are defined, except if your local variable or argument has the same name.

```
public class Person {
        private int age;
        public int getAge() {
                return age;      // returns value of class field age; don’t need to use this.age
        }
        public void setAge(int age) {   // in setters you should assign value of argument to class field
                this.age = age;                 // need to use this.age, because age refers to argument of the function
argument
        }
        public void changeAge (int newAge) {
                age = newAge;                   //age refers to class field; don’t need to use this.age
                int age = newAge + 5;           // age is new local variable, not related to class field age
                This.age = age;                                 // in order to set class field, we need to call it with this.age
now
        }
```

Common mistakes - code (4)

1. Use super to access information from parent class
   a. super(a, b, c); - calls constructor of parent class
   b. super.toString(); - calls parent's class toString() function, so you don't need to copy from there
   c. super.getName(); - to get value of name field defined in parent
   d. this.getName(); - will look for getName() function defined in current class; if there is no such function, will try to find getName() in parent
2. Casting:

```
int a = 5;
~~double b = (double) a;~~ double b = a;          // java will cast for you from int->double, int->long, short -> int
                                                  // any smaller to bigger numerical type
String printStr = "Value is: " ~~+ Integer.toString(a);~~ a;          // when you concatenate string with some other variable
java
                                                  // convert it to string automatically
```

1. Printing: System.out.println(person~~.toString()~~) - println calls toString() for you
2. Avoid empty constructors if you have other constructors