Your code

...in TIG058

Assignment #01

We’ve anonymized your code

Do not take this presentation as we’re saying you’re bad in any way. We bring it up to discuss ….

Sometimes the code ...

Sometimes the code (solution) is way more complicated than the original problem.

Sometimes the code ...

Problem:

Let’s say you have a garage full with cars, parked in numbered slots. How to pick a random car for today’s ride?

Sometimes the code ...

- Would you generate a random license number, loop through the parked cars and check the license numbers hoping to find a match?
- Or would you generate a random slot number and use the car parked in the corresponding slot?
- Any other ideas?

I. e., do you really need to know the license number of the cars in order to pick a random car if the slots are numbered (0, 1, 2 .. (nr-1))?

Something we saw in someone's code

```
private static int [] allowedNodes =
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17,18,19,27,28,29,30,33,34,35,36,37,38,39,41,60,61,62,
63,69,74,75,77,78,109,113,120,200,201,202,203,204,205,206,207,20 ,209,210,211,212,
213,214,245,250};

  public static int GenRandNodeNr () {
    int nodenr = 0;
    Random rand = new Random();
    nodenr = (rand.nextInt(250)) + 1;

    while (Arrays.binarySearch(allowedNodes, nodenr) < 0) {
          nodenr = (rand.nextInt(250)) + 1;
    }
    return (nodenr);
  }

}
```

Average nr of loops?
How many misses? - 191 invalid values if 0…250. A miss percentage 191/250 = 76%
Will it (always) terminate? (It will, but that's thanks to how nextInt is implemented)

Or you could write :)

```
  static Random r = new Random();

  public static int getRandomValue() {
    return allowedNodes[r.nextInt(allowedNodes.length)];
  }
```

Compare previous to the above:

10M calls => 15,517 sec (previous)
10M calls => 1,195 sec (above)

Which version is easiest to understand? To write? To maintain? To review?
Which version is more general? (need to know ordered, length, max/min?)

One idiom we've seen

Can you explain the difference?

==

equals

Misunderstanding Java

Examples of misusing/misunderstanding Java

Misunderstanding Java

```
public class Room implements Comparable <Room>, Comparator <Room> {

   …

}

Collections.binarySearch(list, new Room(rumsNr, 0, 0, 0, 0, "", "")
```

What????? Really what??

Misunderstanding what a comparison can result in

Misunderstanding what a comparison can result in

If you compare Rikard and me with regards to our length, what possible answers could you expect?

- Rikard is taller
- Henrik is taller

More?

- Equal height

Misunderstanding what a comparison can result in

```
public int compare(Room r1, Room r2) {
  if (r1.roomId() < r2.roomId()) {
    return -1;
  } else {
    return 1;
  }
}
```

But, what if two objects ARE equal?

Misunderstanding what the contract dictates

int compare(T o1, T o2)

….

Parameters:
o1 - the first object to be compared.
o2 - the second object to be compared.

Returns:
a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The previous code breaches the contract!

Structure of a method

- Maintainability
- Readability
- Testability
- Correctness and verifiability

Structure of a method

```
    public static void examine(Room r){ //looks for things
      if (!r.getHasBanana() && !r.getThing().equals("No Objects")){
        System.out.printf("\nThere %s %s %s here.\n", getIsAre(r.getThing()), getArticle(r.getThing()).toLowerCase(),
r.getThing(),r.getThing());
      }
      else if (r.getHasBanana() && !r.getThing().equals("No Objects")){
        System.out.printf("\nThere %s %s %s here, and something else... wait... that's the Golden Organic Banana™!\n",
getIsAre(r.getThing()), getArticle(r.getThing()).toLowerCase(), r.getThing(),r.getThing());
      }
      else if (r.getHasBanana() && r.getThing().equals("No Objects")){
        System.out.printf("\nThe Golden Organic Banana™ is here!\n");
      }
      else{
        System.out.print("\nThere is nothing here.\n");
      }
    }
```

Does this scale???
If you have a room, why use a static method instead of asking the room?

Structure of a method

Can you explain what the previous code does?

How long time do you think it took to write the previous code?

How long time do you think it takes to test if the previous code works?

If we add more fruits?

If we add some other things?

If we add other rules?

… let’s look at it again.

Structure of a method

```
public static void examine(Room r){ //looks for things
  if (!r.getHasBanana() && !r.getThing().equals("No Objects")){
    System.out.printf("\nThere %s %s %s here.\n", getIsAre(r.getThing()),
getArticle(r.getThing()).toLowerCase(), r.getThing(),r.getThing());
  }
  else if (r.getHasBanana() && !r.getThing().equals("No Objects")){
    System.out.printf("\nThere %s %s %s here, and something else... wait... that's the
Golden Organic Banana™!\n", getIsAre(r.getThing()), getArticle(r.getThing()).toLowerCase(),
r.getThing(),r.getThing());
  }
  else if (r.getHasBanana() && r.getThing().equals("No Objects")){
    System.out.printf("\nThe Golden Organic Banana™ is here!\n");
  }
  else{
    System.out.print("\nThere is nothing here.\n");
  }
}
```

What if we wrote...

```
public static void examine(Room r){ //looks for things
  System.out.println(r);
}
… and in Room.java
public String toString() {
  return (hasBanana && things.size() == 1) ?
      "You found the banana"
      : (hasBanana && things.size() > 1) ?
        things +  " You found it!"
      : (things.size() > 0) ? things.toString()
      : "No things";
  }
```

```
$ javac Room.java && java TestRoom
empty: No things
things, no banana: [Bird, Cage]
onlyBanana: You found the banana
things and banana: [Banana, Apple] You found it!
```

Or, what if we wrote...

```
public static void examine(Room r){ //looks for things
  System.out.println(r);
}
… and in Room.java
public String toString() {
  String result = things.toString();
  if (things.size() == 0) {
    return "Empty room, where we learned to live …";
  } else if (hasBanana) {
    result += " You found the banana.";
  }
  return result;
}
```

```
$ javac Room.java && java TestRoom
empty: Empty room, where we learned to live …
things, no banana: [Bird, Cage]
onlyBanana: [Banana] You found the banana.
things and banana: [Banana, Apple] You found the banana.
```

Easy to read

How do we make code which is easy to read?

Do you remember?

int compare(T o1, T o2)

….

Parameters:
o1 - the first object to be compared.
o2 - the second object to be compared.

Returns:
a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Which code is easiest to read?

```
Comparator<Double> compDouble = new Comparator<Double>() {
    public int compare(Double s1, Double s2) {
      if (s1 < s2) {
        return -1;
      } else if (s1 > s2) {
        return 1;
      } else {
        return 0;
      }
    }
};
… or, like this?
Comparator<Double> compDouble = new Comparator<Double>() {
    public int compare(Double s1, Double s2) {
       return s1 - s2;
    }
};
.. or,
Comparator<Double> compDouble = (s1, s2) -> { return s1 - s2; };
```

Extra - use of Java 8 constructs (such as streams)

We found this:

```
public static boolean CommandExists (String str) {
  String[] legalCommandsArray =
    {"north", "south", "east", "west","look","examine",
     "pickup","quit","help","n","s","e","w","l","x",
     "p","q","h"};
  Set<String> commandSet =Arrays.stream(legalCommandsArray).collect(Collectors.toSet());
  if (commandSet.contains(str.toLowerCase())) {
    return true;
  } else {
    return false;
  }
}
```

Let’s ignore for now, that methods never have upper case first...

Extra - use of Java 8 constructs (such as streams)

First, return the boolean expression instead:

```
public static boolean commandExists (String str) {
  String[] legalCommandsArray =
    {"north", "south", "east", "west","look","examine",
     "pickup","quit","help","n","s","e","w","l","x",
     "p","q","h"};
  Set<String> commandSet =Arrays.stream(legalCommandsArray).collect(Collectors.toSet());
  return commandSet.contains(str.toLowerCase());
}
```

Extra - use of Java 8 constructs (such as streams)

Second,there are other ways to create a set from an array:

```
public static boolean commandExists (String str) {
  String[] legalCommandsArray =
   {"north", "south", "east", "west","look","examine",
    "pickup","quit","help","n","s","e","w","l","x",
    "p","q","h"};
  Set<String> commandSet = new HashSet<>(Arrays.asList(legalCommandsArray));
  return commandSet.contains(str.toLowerCase());
}
```

But why convert the array to a set?

Extra - use of Java 8 constructs (such as streams)

Third,without an array, you'd have the contains() method already:

```
public static boolean commandExists (String str) {
  List<String> commands = Arrays.asList(
   "north", "south", "east", "west","look","examine",
   "pickup","quit","help","n","s","e","w","l","x",
   "p","q","h");
  return commands.contains(str.toLowerCase());
}
```

Or even:

```
public static boolean commandExists (String str) {
  return Arrays.asList(
   "north", "south", "east", "west","look","examine",
   "pickup","quit","help","n","s","e","w","l","x",
   "p","q","h").contains(str.toLowerCase());
}
```

Java 8 can be powerful, however

We found the following style for reading and adding numbers in books and online:

```
int num, sum;
Scanner sc = new Scanner(new File ("file.txt"));
sum = 0;
while (sc.hasNext()) {
  num = sc.nextInt();
  sum = sum + num;
}
System.out.println("\nSum: " + sum);
```

Java 8 can be powerful, however

With Java 8, you can do this:

```
int sum = Files.lines(Paths.get("file.txt"))
    .mapToInt( i -> Integer.parseInt(i))
    .reduce(0, (x, y) -> { return x + y; } );

System.out.println("\nSum: " + sum);
```