Test the Byte Class

In Lab 1 and Quiz 1 we tested the class `Byte` by means of JUnit test case `ByteTest`.

Question

If we run the JUnit test case `ByteTest` and all tests pass, can we conclude that the class `Byte` correctly implements the API?

Test the Byte Class

In Lab 1 and Quiz 1 we tested the class `Byte` by means of JUnit test case `ByteTest`.

Question

If we run the JUnit test case `ByteTest` and all tests pass, can we conclude that the class `Byte` correctly implements the API?

Answer

No.

Test the Byte Class

In Lab 1 and Quiz 1 we tested the class `Byte` by means of JUnit test case `ByteTest`.

Question

If we run the JUnit test case `ByteTest` and all tests pass, can we conclude that the class `Byte` correctly implements the API?

Answer

No.

Question

Why not?

Test the Byte Class

In Lab 1 and Quiz 1 we tested the class `Byte` by means of JUnit test case `ByteTest`.

Question

If we run the JUnit test case `ByteTest` and all tests pass, can we conclude that the class `Byte` correctly implements the API?

Answer

No.

Question

Why not?

Answer

Run the JUnit test case `ByteTest` several times.

Test the Byte Class

Question

How is it possible that the JUnit test case `ByteTest` passes all tests pass in some runs and fails the method `testIsEven` in other runs?

Test the Byte Class

Question

How is it possible that the JUnit test case `ByteTest` passes all tests pass in some runs and fails the method `testIsEven` in other runs?

Answer

Let's have a look at the code of the method `isEven`.

Test the Byte Class

Question

How is it possible that the JUnit test case `ByteTest` passes all tests pass in some runs and fails the method `testIsEven` in other runs?

Answer

Let's have a look at the code of the method `isEven`.

Answer

Because the method `isEven` uses randomization.

Randomization

Question

Why are we interested in randomization in our code?

Randomization

Question

Why are we interested in randomization in our code?

Answer

The source code of most computer and video games contains some sort of randomization. This provides games with the ability to surprise players, which is a key factor to their long-term appeal.

Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. The MIT Press. 2004.

Randomization

Question

Why are we interested in randomization in our code?

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may reduce the expected running time or memory usage.

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may reduce the expected running time or memory usage.

Question

Which algorithms exploit randomization this way?

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may reduce the expected running time or memory usage.

Question

Which algorithms exploit randomization this way?

Answer

- Randomized quicksort.
- Skiplist.
- ...

Randomization

Question

Why are we interested in randomization in our code?

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may allow us to solve problems.

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may allow us to solve problems.

Question

For which problems is randomization exploited this way?

Randomization

Question

Why are we interested in randomization in our code?

Answer

Randomization may allow us to solve problems.

Question

For which problems is randomization exploited this way?

Answer

- Consensus problem (in an asynchronous distributed system in which processes may fail).
- ...

Nondeterminism

Nondeterministic code is code that, even for the same input, can exhibit different behaviors on different runs, as opposed to deterministic code.

Randomization gives rise to nondeterminism.

Nondeterminism

Nondeterministic code is code that, even for the same input, can exhibit different behaviors on different runs, as opposed to deterministic code.

Randomization gives rise to nondeterminism.

Question

Besides randomization, are there other programming concept that give rise to nondeterminism?

Nondeterminism

Nondeterministic code is code that, even for the same input, can exhibit different behaviors on different runs, as opposed to deterministic code.

Randomization gives rise to nondeterminism.

Question

Besides randomization, are there other programming concept that give rise to nondeterminism?

Answer

Concurrency.

Concurrency in Java

A Crash Course

www.cse.yorku.ca/course/4315/

Books

- Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes and Doug Lea. Java Concurrency in Practice. Addison-Wesley, 2006.
- Mary Campione, Kathy Walrath and Alison Huml. The Java Tutorial. Lesson: Threads: Doing Two or More Tasks At Once.
- James Gosling, Bill Joy, Guy L. Steele Jr. and Gilad Bracha. The Java Language Specification. Third edition.

Thread Creation

In Java, threads are created dynamically:

```
// create and initialize Thread object
Thread thread = new Thread();
// execute run method of Thread object concurrently
thread.start();
```

The class `Thread` is part of package `java.lang` (and, hence, does not need to be imported). Its API can be found at the URL `https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html`.

Thread API

- `public Thread(String name)`
  Initializes a new Thread object with the specified name as its name.
- `public void start()`
  Causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread.
- `public void run()`
  This method does nothing and returns.

Printer

Question

Develop a Java class called `Printer` that is a `Thread` and prints its name 1000 times.

Two Concurrent Printers

Question

Develop an app that creates two `Printer`s with names 1 and 2 and run them concurrently.

Two Concurrent Printers

Question

Develop an app that creates two `Printer`s with names 1 and 2 and run them concurrently.

Question

What is the output of the app?

Two Concurrent Printers

Question

Develop an app that creates two `Printer`s with names 1 and 2 and run them concurrently.

Question

What is the output of the app?

Answer

A sequence of 1000 1's and 2's (arbitrarily interleaved). This example shows that concurrency gives rise to nondeterminism.

Two Concurrent Printers

Question

What happens if we replace start with run in the app?

Two Concurrent Printers

Question

What happens if we replace start with run in the app?

Question

Let's try it.

Two Concurrent Printers

Question

What happens if we replace start with run in the app?

Question

Let's try it.

Question

The output is a sequence of 1000 1's followed by 1000 2's.

Java only Supports Single Inheritance

The following is not allowed in Java.

```
public class Printer extends Applet, Thread
```

Thread Creation

```
// create and initialize Runnable object
Runnable runnable = new ...();
// create and initialize Thread object
Thread thread = new Thread(runnable);
// execute run method of Runnable object concurrent
thread.start();
```

The interface `Runnable` is part of package `java.lang` (and, hence, does not need to be imported). Its API can be found at the URL `https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html`.

Runnable is an Interface

In Java, you cannot create instances of an interface.

```
public class Printer implements Runnable
{
   ...
}
```

The assignment

```
Runnable printer = new Printer();
```

is valid since the class `Printer` implements the interface `Runnable`.

Printer

Question

Develop a Java class called `Printer` that implements `Runnable` and prints the thread's name 1000 times.