Java 7 features

Agenda

- Core language features
- NIO2
- Fork and Join
- java.util.concurrent.Phaser

String in switch statements

```
switch (parameter) {
case "test1" :
    return "value was test1";
case "test2" :
    return "value was test2";
default:
    return "default value";
}
```

Easier resource management

```
public static String readLine(String path) throws Exception {
    // resources have to implement AutoCloseable interface
    try (
        FileReader fr = new FileReader(path);
        BufferedReader br = new BufferedReader(fr)
    ) {
        return br.readLine();
    }
}
```

- suppressed exceptions (Throwable.getSuppressed();)

Multiple exceptions handling

possibility to catch more than one exception in single catch block

```
catch (FileNotFoundException | IllegalAccessException e) {
    ...
}
```

Diamond operator

No need to explicitly write all generic class parameters

```
Map<String, List<Long>> map = new HashMap<>();

List<String> originalInitialization = new ArrayList<String>();

List<String> newInitialization = new ArrayList<>();
```

NIO2 - java.nio.*

New abstractions for work with folders/files

- Path
  - abstraction for work with directories/files
- Files
  - helper static methods for directory/file operations
- FileStore
  - Files.getFileStore(Path path)
  - information about the storage (total space, usable space etc.)

NIO2

- FileSystem
  - Operations on file system, eg. returning Path instance based on name
    - getPath(“/foo/bar”)
- FileSystems
  - Access to default FileSystem
  - Possibility to create custom FileSystem instances
- Async file channels

NIO2 - create file

```
Path target = Paths.get("tmp/file.txt");

Path file1 = Files.createFile(target);



Set<PosixFilePermission> perms = PosixFilePermissions.
fromString("rw-rw-rw-");
FileAttribute<Set<PosixFilePermission>> attr =
PosixFilePermissions.asFileAttribute(perms);

Path file2 = Files.createFile(target, attr);
```

NIO2 - delete file

```
Path target = Paths.get("/tmp/example.txt");
Files.delete(target);
```

NIO2 - copy file

```
Path source = Paths.get("/tmp/file.txt");
Path target = Paths.get("/tmp/file2.txt");

Files.copy(source, target);
```

or

```
Files.copy(source, target, StandardCopyOption.
REPLACE_EXISTING);
```

NIO2 - move file

```
Path source = Paths.get("/tmp/file.txt");
Path target = Paths.get("/tmp/file2.txt");

Files.move(source, target,
StandardCopyOption.REPLACE_EXISTING,
StandardCopyOption.COPY_ATTRIBUTES);
```

NIO2 - WatchService

monitor folder for changes

- create and register WatchService
  - specify events you want to catch
- start loop to catch events
- consume WatchKey and process event

Fork and Join

- Easily split tasks among more threads and collect results back
- see example

java.util.concurrent.Phaser

barrier waiting until tasks in one batch finish and then lets them go again

might look like CyclicBarrier/CountDownLatch but lets you unregister parties in runtime

Links

All examples and links on https://www.github.com/sitina/java7features