Removed if-statements

Calc.getRandomIntBetween()

Removed a double If-statement that checked boundry values into one if-statement.
See: `git diff a0f05d2 2672a38`

VerticalPlatform.update() & HorizontalPlatform.update()

Beforehand, there even was an additional method updateEnums() with one if-statement which has moved into the update method and has been modified to remove all pre-existing if-statements in the update() method. This is because the platform is now self-aware of its speed, rendering the whole system present before useless and making the if-statements it used absolete.
See: `git diff 2672a38 4128f2c`

FileSystem.deleteFile()

There was a if in the deleteFile() method that checked if deleting the file was successfull and only logged something in that case. Now the filesystem will always log something, indicating whether or not deleting the file was succesfull.
See: `git diff b125ab2f 20c0949`

DoodleCamera.update()

The update method in the DoodleCamera previously had an if-statement checking if the camera position must be updated, now we use a little bit more complex maths to determine the new position without an if-statement.
See: `git diff d7c6f97 2ea775e`

AudoManager.Sound.play()

There was an if-statement which checked if there was actually a clip runnen and stopped that clip. This check is not necessary because the clip can always be stopped, whether it is playing or not.
See: `git diff 1e80cfc b14b80a`

SpriteFactory.getDigitSprite(...)

Contained one if-statement and a switch-statement. However, the if-statement performed a check that was included in the switch-statement.
See: `git diff 0d9a04b 800ef86`

Also removed the switch-statement which remained after the refactor described above. This switch-statement was used to return a specific number. Now the method will just return an array

of ISprites containing all digits. The only change when using this method is that the index given to the method beforehand must now be appended to the call as [index] (to select the index in the array).
See: `git diff 800ef86 c98118d`

SpriteFactory.getDoodleSprite(...)

Contained a if-statement making a destinction between a right-facing doodle and a left-facing doodle. These have now be implemented in seperate methods. The implementation using this method was already written for this change.
See: `git diff d109acc a70fef8`

SpriteFactory.loadISprite(...)

This method had a redundent if-statement that returned something when a certain variable was null, but this was already covered by the try-catch block put in place to catch the exception which would cause the null-value.
See: `git diff 326fccd 9369168`

KillScreen.render()

Contained an if-statement from an old implementation rendering the scene only conditionally. The current implementation of the game doesn’t require this as the scene doesn’t exist when it shouldn’t be active.
See: `git diff dcdbac4 2c8bdd0`

PauseScreen.render()

Contained an if-statement from an old implementation rendering the scene only conditionally. The current implementation of the game doesn’t require this as the scene doesn’t exist when it shouldn’t be active.
See: `git diff 215d08c 8ac8673`

World.ScoreBar.ScoreText.getHighestScore()

Contained an if-statement comparing two numbers, now using Math.max().
See: `git diff ab2e0fd 21db308`

World.ScoreBar.PauseButton.mouseClicked()

Contained an if-statement, however the whole button could be implemented using the normal Button implementation.
See: `git diff 21db308 ad92e55`

HighScoreList.initHighScores()

Contained an if-statement checking for null-values, the method which could potentially return the null-value now returns an empty string in these cases.
See: `git diff 88eb3ae 5f94c20`

HighScoreList.loadFromFile()

Contained in if-statement checking the size of the list received from reading the file. This will now be handled by the (already present) try-catch block. As a result, an empty string will be returned so the game can continue.
See: `git diff 24e3b5e d5285d0`

Remaining if-statements

Button.mouseClicked()

This method contains an if-statement checking if the click was within certain boundries, removing this would make the code unnecessarily complex.

ButtonFactory.getButtonFactory()

Contains one if-statement for a singleton implementation which we want to maintain.

FileSystem.getResourceFile()

Contains three if-statements, two of which ensure there are no null-values which we want to maintain as such because the method is public. And changing the other one, which prepends a character conditionally, would make the code unnecessarily complex.

Logger.appendToTextFile()

Contains one if-statement which ensures that the logger only logs when it should log (pending tasks).

LoggerFactory.LoggerFactory()

Contains one if-statement which makes sure the log file is being cleared only when a certain value is true. Removing this would make the code unnecessarily complex.

LoggerFactory.createLogger()

Contains one if-block that makes sure that classes that shouldn't log, don't log by returning a fake logger (an implementation of ILogger without any actual code). Removing this would only introduce more if-statements in each class that uses a Logger.

Calc.getRandomIntBetween()

Contains one if-statement which ensures the input values are correct. Removing this would make using the method unsafe. And replacing with an assert is not recommended because the method is public.

Calc.getRandomDouble()

Contains one if-statement which ensures the input value is correct. Removing this would make using the method unsafe. And replacing with an assert is not recommended because the method is public.

Renderer.drawSprite(...), Renderer.drawSpriteHUD(...)

All contain one if-statement which ensures the input sprite is not null. Removing this would make using the method unsafe. And replacing with an assert is not recommended because the method is public.

Renderer.setGraphicsBuffer(...)

Cotains one if-statement which ensures the given graphics buffer is not null. Removing this would make using the method unsafe. And replacing with an assert is not recommended because the method is public.

[ANY CLASS].register() - provided it has a register method

Every register method, used by our service locator, has an if statement checking if the service locator isn't null. Even though the Game would not work at all when the register method is invoked incorrectly, because the game requires all object provided by the service locator to work and it would be clear that something is wrong. We feel it is better to have a if-statement checking for this condition and throwing a (known) error, so it is easier to debug.
It is important to note that the if-statement in each register method is called at most one time, when the game is launched.

HighScore.compareTo(...)

Should return a non-boolean value based on a boolean result, implementation with if-statements is the best solution.

HighScore.equals(...)

A good implementation of equals() must contain an if-statement checking if the given object is null (because it will give a nullpointerException when looking at the attributes). Besides that we use them for early outs (the given object is the current object; Classes are not the same) improving the efficiency.

Ignored

Res.setSkin()
Menu.update()
[Any class].keyRelease() - Changing this will change a large portion of the code base