**2.1 Naming conventions**

**2.1.1 Variable Names**

2.1.1.1 Letters in local variables should start with a lower case letter.

2.1.1.2 Variables formed with more than one word should have the first letter

of each word except for the first word capitalized.

2.1.1.3 Variable names ‘I’, ’j’, ‘k’ should be used only in iterations. E.g. in a for loop.

2.1.1.4 Private or protected variable names begin with “my.”

2.1.1.5 Static private or protected variable names begin with “our.”

2.1.1.6 If the method reads the parameter but does not modify its contents, prefix the parameter with in. If the method receives an empty parameter and sets its contents as a return value, prefix the parameter with out. If the method reads the contents of a parameter and modifies them, prefix the parameter with inOut.

2.1.1.7 Letters in global variables should all be capitalized.

**Example:**

int variable

int myVariableName

int GLOBALVARIABLE

**2.1.2 Function Names**

2.1.2.1 The first letter of private and public function should be lower case.

2.1.2.2 Functions formed with more than one word should have the first letter

of each word except for the first word capitalized.

2.1.2.3 Method that reports on objects of type X should be named “getX.”

2.1.2.4 Factory method that creates objects of type X should be named “newX.”

2.1.2.5 Method that changes value of objects of type X should be named “setX.”

2.1.2.6 Method that returns attribute Y with different data type X should be named “toX.”

**Example:**

public int value()

public int setValue()

public String toString()

**2.1.3 Class Names**

2.1.3.1 The first letter of private and public class should be capitalized.

2.1.3.2 Classes formed with more than one word should have the first letter

of each word except for the first word capitalized. (2.1.3.1 overrides.)

2.1.3.3 Exception class name ends with “Exception.”

2.1.3.4 When necessary to distinguish class from similarly named interfaces, class name ends with “Impl.”

**Example:**

public class Light

public class TrainControlImpl

**2.1.4 Constants**

2.1.4.1 All letters in a constant name should be capitalized.

2.1.4.2 Constants with more than one word should be separated using an underscore.

**Example:**

public static final int BLADE=10;

public static final int TRAFIC\_LIGHT=20;

**2.1.5 Interface**

2.1.5.1 Interface should follow classes’ naming convention. However, before the first letter, the name should have a capital letter ‘I’ attached.

**Example:**

interface IModle

**2.1.6 Packages**

2.1.6.1 Letters in packages’ names should start with a lower case letter.

2.1.6.2 Before the first letter, the name should have a the word “SHWOZ” attached.

Example:

SHWOZspirit.jar

**2.2 Comment**

**2.2.1 Block comments**

2.2.1.1 Block comments should use javadoc formatting.

2.2.1.2 Each file should have a block comment at the top of the file. The block

comment should include: file name, author, group name, and date.

2.2.1.3 Each function should have a block comment before the function name

line. The block comment should include: a brief description of the function,

input parameters, and output parameters.

2.2.1.4 Each class should have a block comment before the class name line. The

block comment should include: a brief description of the class, the function it

includes.

**Example:**

/\*This class is for controlling the lights

Function included: lightsOn, lightsOff\*/

**2.2.2 Line comments**

2.2.2.1 Line comments should be added when the name and-or purpose of a

variable is not very clear for the users

2.2.2.2 Line comments should be added right after the line of code is applied, the line comment and the code should be separated by one tab

2.2.2.3 If the line comment is too long to hold in the same line as the code, add the line comment before the line of code

**Example:**

x=y+z; //calculate the value of x

// The following variable shows the control value when train is on yellow line

Int trainControlYellow=0;

**2.3 Indentation**

**2.3.1 Spaces**

2.3.1.1 Spaces should be added between mathematical/logical operators and the

Operands

**Example:**

while (x == 2)

{

y = z + 1;

}

**2.3.2 Tabs**

2.3.2.1 Tabs should be formed in the width of 4 spaces

2.3.2.2 Tabs should be used to indent lines.

2.3.2.2 Each statement inside a block should be one tab from the start brace

**Example:**

if (x != 1)

{

x=1;

}

**2.4 Code formatting and practices**

**2.4.1 Braces**

2.4.1.1 The opening and closing braces of a statement should have its own line

2.4.1.2 Braces should always be there, even if the functions or loop blocks

only contain one line

**Example:**

if(!here)

{

return;

}

**2.4.2 Statements**

2.4.2.1 There should only be one statements per line

2.4.2.2 There should only be one variable declared per line

2.4.2.3 ‘==true’ and ‘==false’ should not be used.

2.4.2.4 A ‘default’ case should always be included in a switch statement even if it’s not

needed.

**Example:**

while(flag)

if(!flag)

int here;

int there;

switch(name)

{

case 1:

…

break;

case 2:

…

break;

default:

break;

}

**2.4.3 Exceptions**

2.4.3.1 Operations that has a possibility to generate exceptions should be contained

by a try/catch block. The catch block should indicate the exception information