You are expected to follow these Java coding standards when writing Java code this semester. Every time a standard is not adhered to, the reason must be clearly documented.

Please note that the examples shown in this document is not for code which can be compiled. **They are meant for illustrations of the coding standards only**. For code syntax please refer to the lectures and tutorials.

1. General Code Layout

A. Using Spaces

The use of spaces needs to be justified and consistent to ensure high quality code. The following are some of the key uses of spaces which must be adhered to.

Spaces around operators

An operator in Java essentially links two variables and/or keywords or operands. Operations can be arithmetic (+, -, *, /, % etc.), relational (>, >=, <, <=, !=, etc.), conditional (&&, ||, etc.) or assignment (=). Note the following examples, one demonstrates the correct use of spaces, the other an incorrect use of spaces.

Correct Use of Space	Incorrect Use of Spaces

`/**`
` * An example of a method`
` *`
` * @param  y  a sample parameter for a method`
` * @return  the sum of x and y`
` */`
`public int sampleMethod(int y)`
`{`
`  int num1 = 0;`
`  int num2 = 0;`
`  return num1 + num2;`
`}`
	`/**`
	` * An example of a method`
	` *`
	` * @param  y  a sample parameter for a method`
	` * @return  the sum of x and y`
	` */`
	`public int sampleMethod(int y)`
	`{`
	`  int num1 =0;`
	`  int num2=0;`
	`  return num1  + num2  ;`
	`}`

Spaces around conditional statements and loops

A conditional statement in Java specifies an IF or ELSE condition. Loops allow code to repeat until a condition is satisfied. Note the following examples, and more important the placement of spaces between them.

- Use a space between an if, switch, while, for, and do while loops, and the following parenthesis. For example:

Correct Use of Space	Incorrect Use of Spaces

```java
/**
 * An example of a method - replace this comment with your own
 *
 * @param  y   a sample parameter for a method
 * @return     the sum of x and y
 */
public int sampleMethod(int y)
{
    int num1 = 0;
    int num2 = 0;
    if (num1 >= 0)
        num2 = num1 + 1;
    for (int i = 0; i < 10; i++)
    {
    }
    int i = 0;
    while (i < 10)
    {
        i++;
    }
    do
    {
    } while (i < 10);
    switch (y)
    {
        case 1:
            num1++; break;
        case 2:
            if(y > num1)
            {
                num2++;
                num1--;
            }
            break;
        default:
            y++;
            break;
    }
    return num1 + num2;
}
```

```java
/**
 * An example of a method - replace this comment with your own
 *
 * @param  y   a sample parameter for a method
 * @return     the sum of x and y
 */
public int sampleMethod(int y)
{
    int num1 =0;
    int num2=0;
    if(num1>=0)
        num2 = num1+1;
    for(int i = 0;i < 10;i++)
    {
    }
    int i=0;
    while(i<10)
    {
        i++;
    }
    do
    }while(i<10);
    switch(y)
    {
        case 1:
        num1++;
        break;
        case 2:
        if(y > num1)
        {
            num2++;
            num1--;
        } break;
        default:
        y++;
        break;
    }
    return num1 + num2;
}
```

Use a space between the operators within an if, switch, while, for, and do while loops. For example:

Correct Use of Space	Incorrect Use of Spaces

```java
/**
 * An example of a method - replace this comment with your own
 *
 * @param  y   a sample parameter for a method
 * @return     the sum of x and y
 */
public int sampleMethod(int y)
{
    int num1 = 0;
    int num2 = 0;
    if (num1 >= 0)
        num2 = num1 + 1;
    for (int i = 0; i < 10; i++)
    {
    }
    int i = 0;
    while (i < 10)
    {
        i++;
    }
    do
    {
    } while (i < 10);
    switch (y)
    {
        case 1:
            num1++; break;
        case 2:
            if(y > num1)
            {
                num2++;
                num1--;
            }
            break;
        default:
            y++; break;
    }
    return num1 + num2;
}
```

```java
/**
 * An example of a method - replace this comment with your own
 *
 * @param  y   a sample parameter for a method
 * @return     the sum of x and y
 */
public int sampleMethod(int y)
{
    int num1 =0;
    int num2=0;
    if(num1>=0)
        num2=num1+1;
    for(int i = 0;i < 10;i++)
    {
    }
    int i=0;
    while(i<10)
    {
        i++;
    }
    do
    {
    }while(i<10);
    switch(y)
    {
        case 1:
        num1++;
        break;
        case 2:
        if(y > num1)
        {
            num2++;
            num1--;
        } break;
        default:
        y++;
        break;
    }
    return num1 + num2;
}
```

Spaces around internal and external method calls, and placement of statements

Methods allow code to be written so as to maximize reusability. Internal method calls are methods which exist in the same class. External method calls are methods which exist in another class. Note the following examples, and more important the placement of spaces between them.

- Do not use a space before the parenthesis in a method call. For example:

Correct Use of Space

```java
/**
 * An example of a method - replace this comment with your own
 *
 * @param  y   a sample parameter for a method
 * @return     the sum of x and y
 */
public int sampleMethod(int y)
{
    int id = getId(y);
    Person per = new Person(id);
    per.getName();
}
```

Incorrect Use of Spaces

```java
/**
 * An example of a method - replace this comment with your own
 *
 * @param  y   a sample parameter for a method
 * @return     the sum of x and y
 */
public int sampleMethod(int y)
{
    int id = getId (y);
    Person per = new Person(y);
    per.getName ();
}
```

- Place each statement on a separate line. An exception is in a switch statement where a break may be placed on the same line as the preceding statement.

Correct Placement of Statements	Incorrect Placement of Statements

`/**`
` * An example of a method - replace this comment with your own`
` *`
` * @param  y  a sample parameter for a method`
` * @return  the sum of x and y`
` */`
`public int sampleMethod(int y)`
`{`
`  int id = getId(y);`
`  Person per = new Person(id);`
`  per.getName();`
`}`
	`/**`
	` * An example of a method - replace this comment with your own`
	` *`
	` * @param  y  a sample parameter for a method`
	` * @return  the sum of x and y`
	` */`
	`public int sampleMethod(int y)`
	`{`
	`  int num1 =0;`
	`  int num2=0;`
	`  if(num1>=0)`
	`  num2 = num1+1;`
	`  for(int i = 0;i < 10;i++)`
	`  {`
	`  }`
	`  int i=0;`
	`  while(i<10)`
	`  {`
	`  i++;`
	`  }`
	`  do`
	`  {`
	`  }while(i<10);`
	`  switch(y)`
	`  {`
	`  case 1:`
	`  num1++;`
	`  break;`
	`  case 2:`
	`  if(y > num1)`
	`  {`
	`  num2++; num1--;`
	`  } break;`
	`  default:`
	`  y++;`
	`  break;`
	`  }`
	`  return num1 + num2;`
	`}`

Define and initialise each variable or field (attribute) on a separate line. For example:

Correct Declaration and Initialisation of Variables	Incorrect Declaration and Initialisation of Variables

`/**`
` * An example of a method - replace this comment with your own`
` *`
` * @param  y  a sample parameter for a method`
` * @return  the sum of x and y`
` */`
`public int sampleMethod(int y)`
`{`
`  int id;`
`  String name;`
`  int age = 10;`
`  String phoneNo = "1234";`
`}`
	`/**`
	` * An example of a method - replace this comment with your own`
	` *`
	` * @param  y  a sample parameter for a method`
	` * @return  the sum of x and y`
	` */`
	`public int sampleMethod(int y)`
	`{`
	`  int id, age = 10;`
	`  String name, phoneNo = "1234";`
	`}`

- Place a blank line between methods (and constructors).

Correct Separation Between Methods	Incorrect Separation Between Methods

```java
/**
 * Write a description of class Temp here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Temp
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Temp
     */
    public Temp()
    {
        // initialise instance variables
        x = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int sampleMethod(int y)
    {
        int id;
        String name;

        int age = 10;
        String phoneNo = "1234";
    }
}
```

```java
/**
 * Write a description of class Temp here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Temp
{
    // instance variables - replace the example below with your own
    private int x;
    /**
     * Constructor for objects of class Temp
     */
    public Temp()
    {
        // initialise instance variables
        x = 0;
    }
    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int sampleMethod(int y)
    {
        int id;
        String name;

        int age = 10;
        String phoneNo = "1234";
    }
}
```

Layout of Class Declaration

- Place each class in a separate file.

Correct Declaration of Classes	Incorrect Declaration of Classes

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Finish
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }
}
```

The elements of a class should be declared in the following order:

- import statements
- class comment
- class header
- field definitions
- constructors
- methods

Correct Class Skeleton	Incorrect Class Skeleton

```java
import java.util.Scanner;
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int sampleMethod(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
import java.util.Scanner;
public class Start
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int sampleMethod(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }
}
```

- Methods (excluding constructors) in a class declaration should be arranged in alphabetical order of the name of method.

Correct Ordering of Methods	Incorrect Ordering of Methods

```java
public class Start
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int sampleMethod(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int sampleMethodTwo(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

```java
public class Start
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int sampleMethodTwo(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int sampleMethod(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

Brackets and Indenting

- Use a consistent level of indentation. One tab or four spaces is suggested.
- Indent your code one level per block (pair of braces).
- Vertically align matching braces. (Note that this convention differs from that used in the textbook Objects First with Java, Barnes & Kölling.)

Correct Brackets and Indenting	Incorrect Brackets and Indenting

```java
/**
 * An example of a method - replace this comment with your own
 *
 * @param  y   a sample parameter for a method
 * @return     the sum of x and y
 */
public int sampleMethod(int y)
{
    int num1 = 0;
    int num2 = 0;
    if (num1 >= 0)
        num2 = num1 + 1;
    for (int i = 0; i < 10; i++)
    {
        y = num1 + num2;
    }
    int i = 0;
    while (i < 10)
    {
        i++;
    }
    do
    {
        i++;
    } while (i < 10);
    switch (y)
    {
        case 1:
            num1++; break;
        case 2:
            if(y > num1)
            {
                num2++;
                num1--;
            }
            break;
        default:
            y++; break;
    }
    return num1 + num2;
}
```

```java
/**
 * An example of a method - replace this comment with your own
 *
 * @param  y   a sample parameter for a method
 * @return     the sum of x and y
 */
public int sampleMethod(int y)
{
    int num1 =0;
    int num2=0;
    if(num1>=0)
        num2 = num1+1;
    for(int i = 0;i < 10;i++) {
        y = num1 + num2;
    }
    int i=0;
    while(i<10) {
            i++;
    }
    do
    {
        i++;
    }while(i<10);
    switch(y)
    {
        case 1:
        num1++;
        break;
        case 2:
        if(y > num1)
        {
            num2++; num1--;
        } break;
        default:
        y++;
        break;
    }
    return num1 + num2;
}
```

- Use braces in if .. else and loop constructs only when necessary to define blocks of code with more than one statement. Note that this convention differs from that used in the textbook Objects First with Java, Barnes & Kölling.

Correct Brackets and Indenting	Incorrect Brackets and Indenting

```java
/**
 * An example of a method - replace this comment with your own
 *
 * @param  y   a sample parameter for a method
 * @return     the sum of x and y
 */
public int sampleMethod(int y)
{
    int num1 =0;
    int num2=0;
    if(num1>=0)
        num2 = num1+1;
    for(int i = 0;i < 10;i++) {
        y = num1 + num2;
    }
    int i=0;
    while(i<10) {
            i++;
    }
    do
    {
        i++;
    }while(i<10);
    switch(y)
    {
        case 1:
        num1++;
        break;
        case 2:
        if(y > num1)
        {
            num2++; num1--;
        } break;
        default:
        y++;
        break;
    }
    return num1 + num2;
}
```

```java
 * An example of a method - replace this comment with your own
 *
 * @param  y   a sample parameter for a method
 * @return     the sum of x and y
 */
public int sampleMethod(int y)
{
    int num1 =0;
    int num2=0;
    if(num1>=0)
    {
        num2 = num1+1;
    }
    for(int i = 0;i < 10;i++) {
        y = num1 + num2;
    }
    int i=0;
    while(i<10) {
            i++;
    }
    do
    {
        i++;
    }while(i<10);
    switch(y)
    {
        case 1:
        num1++;
        break;
        case 2:
        if(y > num1)
        {
            num2++;
            num1--;
        } break;
        default:
        y++;
        break;
    }
    return num1 + num2;
}
```

- Align second and additional conditions for if statements that run over multiple lines under the first condition after the parenthesis. Align arguments in method calls that run over multiple lines under the first argument. Take care that no code extends past the side of the screen or page. (The maximum line length should be between 80 and 100 characters in length)

Correct Code Alignment	Incorrect Code Alignment

```java
public int sampleMethod(int y)
{
    int num1 = 0;
    int num2 = 0;
    if (num1 >= 0)
        num2 = num1 + 1;
    for (int i = 0; i < 10; i++)
    {
        y = num1 + num2;
    }
    int i = 0;
    while (i < 10)
    {
        i++;
    }
    do
    {
        i++;
    } while (i < 10);
    switch (y)
    {
        case 1:
            num1++; break;
        case 2:
            if(((y > num1) && (y > num2)) ||
                ((y < num1) && (y < num2))
            {
                num2++;
                num1--;
                System.out.println("The maximum line length should be between 80 - 100 characters only!!");
            }
            break;
        default:
            y++; break;
    }
    return num1 + num2;
}
```

```java
public int sampleMethod(int y)
{
    int num1 =0;
    int num2=0;
    if(num1>=0)
        num2 = num1+1;
    for(int i = 0;i < 10;i++) {
        y = num1 + num2;
    }
    int i=0;
    while(i<10) {
            i++;
    }
    do
    {
        i++;
    }while(i<10);
    switch(y)
    {
        case 1:
        num1++;
        break;
        case 2:
        if(((y > num1) && (y > num2)) || ((y < num1) && (y < num2))
        {
            num2++;
            num1--;
            System.out.println("The maximum line length should be between 80 - 100 characters only!!");
        } break;
        default:
        y++;
        break;
    }
    return num1 + num2;
}
```

Declaring and Initialising Variables

- Initialise all variables (except fields in a class declaration) when they are created.

Correct Field Initialisation	Incorrect Field Initialisation
`/** * Write a description of class Start here. * * @author (your name) * @version (a version number or a date) */ public class Start { // instance variables - replace the example below with your private int x; /** * Constructor for objects of class Start */ public Start() { // initialise instance variables x = 0; } }`	`/** * Write a description of class Start here. * * @author (your name) * @version (a version number or a date) */ public class Start { // instance variables - replace the example below with your own private int x = 0; /** * Constructor for objects of class Start */ public Start() { // initialise instance variables } }`

- Explicitly define all fields and methods as private, protected or public.

Correct Visibility Modifier Definition	Incorrect Visibility Modifier Definition

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int sampleMethodTwo(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
class Start

    // instance variables - replace the example below with your own
    int x = 0;

    /**
     * Constructor for objects of class Start
     */
    Start()

        // initialise instance variables
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    int sampleMethodTwo(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

- Fields should not be specified as public (except for final fields).

Correct Field Visibility Modifier Definition	Incorrect Field Visibility Modifier Definition

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your
    private int x;
    public static final VALUE = 10;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your own
    public int x;
    public static final VALUE = 10;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }
}
```

- Constant fields should be declared as static.

Correct Declaration of Constants	Incorrect Declaration of Constants

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your own
    public int x;
    public static final VALUE = 10;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your own
    private int x;
    public final VALUE = 10;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }
}
```

Naming

- Use descriptive and meaningful names for all identifiers (names of classes, methods and variables). Avoid ambiguous names. Avoid abbreviations.

Correct Naming Conventions	Incorrect Naming Conventions

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class StartGame
{
    // instance variables - replace the example below with your own
    private int valueX;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        valueX = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  newValue   a sample parameter for a method
     * @return     the sum of valueX and newValue
     */
    public int calculateSum(int newValue)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int method1(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

- Use the convention whereby names with multiple words are joined and have uppercase letters at the start of the second and successive words. This is sometimes called camel case. Do not use underscore, minus or punctuation characters to separate words.

Correct Naming Conventions	Incorrect Naming Conventions

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int method1(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class StartGame
{
    // instance variables - replace the example below with your own
    private int valuex;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        valueX = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  newValue   a sample parameter for a method
     * @return     the sum of valueX and newValue
     */
    public int calculate_sum(int newvalue)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

- Class names start with a capital letter and have an uppercase letter starting every word in their name. Java requires that a class be stored in a file with the same name as the class and with a .java extension (e.g: CreditCardAccount.java).

Correct Class Naming Conventions	Incorrect Class Naming Conventions
`/** * Write a description of class Start here. * * @author (your name) * @version (a version number or a date) */ public class Start { // instance variables - replace the example below with your own private int x; /** * Constructor for objects of class Start */ public Start() { // initialise instance variables x = 0; } /** * An example of a method - replace this comment with your own * * @param y a sample parameter for a method * @return the sum of x and y */ public int method1(int y) { int id = getId(y); Person per = new Person(id); per.getName(); } }`	`/** * Write a description of class Start here. * * @author (your name) * @version (a version number or a date) */ public class startgame { // instance variables - replace the example below with your own private int valuex; /** * Constructor for objects of class Start */ public Start() { // initialise instance variables valueX = 0; } /** * An example of a method - replace this comment with your own * * @param newValue a sample parameter for a method * @return the sum of valueX and newValue */ public int calculate_sum(int newvalue) { int id = getId(y); Person per = new Person(id); per.getName(); } }`

- Method names start with a lowercase letter and the first letter of each word after that is in uppercase.

Correct Method Naming Conventions	Incorrect Method Naming Conventions

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Start
{
    // instance variables - replace the example below with your own
    private int x;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        x = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y   a sample parameter for a method
     * @return     the sum of x and y
     */
    public int method1(int y)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class startgame
{
    // instance variables - replace the example below with your own
    private int valuex;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        valueX = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  newValue   a sample parameter for a method
     * @return     the sum of valueX and newValue
     */
    public int Calculatesum(int newvalue)
    {
        int id = getId(y);
        Person per = new Person(id);
        per.getName();
    }
}
```

- Object and variable names start with a lowercase letter and the first letter of each word after that is in uppercase (the same as for methods).

Correct Object and Variable Naming Conventions	Incorrect Object and Variable Naming Conventions

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class StartGame
{
    // instance variables - replace the example below with your own
    private int valueX;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        valueX = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  newValue   a sample parameter for a method
     * @return     the sum of valueX and newValue
     */
    public int calculateSum(int newValue)
    {
        int id = getId(y);
        Person personOne = new Person(id);
        personOne.getName();
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class startgame
{
    // instance variables - replace the example below with your own
    private int valuex;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        valueX = 0;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  newValue   a sample parameter for a method
     * @return     the sum of valueX and newValue
     */
    public int Calculatesum(int newvalue)
    {
        int id = getId(y);
        Person personone = new Person(id);
        per.getName();
    }
}
```

- Constants (or final variables) are written in uppercase with underscores (_) to separate words.

Correct Constant Naming Conventions	Incorrect Constant Naming Conventions

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class StartGame
{
    // instance variables - replace the example below with your own
    private static final int MAX_VALUE = 10;
    private int valueX;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        valueX = 0;
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class StartGame
{
    // instance variables - replace the example below with your own
    private static final int maxvalue = 10;
    private int valueX;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        valueX = 0;
    }
}
```

Documentation

- Place a comment at the top of each method and class definition to explain its purpose.

Correct Documentation Placement

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class StartGame
{
    // instance variables - replace the example below with your own
    private static final int maxvalue = 10;
    private int valueX;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        valueX = 0;
    }
}
```

Incorrect Documentation Placement

```java
public class StartGame
{
    // instance variables - replace the example below with your own
    private static final int maxvalue = 10;
    private int valueX;

    public Start()
    {
        // initialise instance variables
        valueX = 0;
    }
}
```

- Include comments in code only when necessary to explain complex code. Note that, where possible, it is preferable to write clear comprehensible code.
- Any comments written INSIDE a method to clarify the code should be written as in the example below.
- Alternatively, if the comment will not fit on one line, as in the previous example, write the comment on a line of its own, as follows:

Correct Comment Placement	Incorrect Comment Placement

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class StartGame
{
    private static final int maxvalue = 10; //Maximum value supported
    private int valueX;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        // initialise instance variables
        valueX = 0;
    }
}
```

```java
/**
 * Write a description of class Start here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class StartGame
{
    //Maximum value supported
    private static final int maxvalue = 10;
    private int valueX;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        valueX = 0; // initialise instance variables
    }
}
```

- BlueJ will create a skeleton comment at the top of the class, which you are to fill out. The comment should include (at least) the purpose of the class, the author(s) and the date created. Every person who contributed to the class must be named as an author or otherwise appropriately acknowledged.
- Class comments must be recognised by Javadoc. In other words, they should start with the comment symbol /**
- For each extra line of comment you include, you must add * to the start of the line. The * is to be preceded by one space and followed by one space before the start of that line of the comment.
- See the textbook Objects First with Java, Barnes & Kölling for further details of Javadoc and other options available.

Correct Documentation Inclusions	Incorrect Documentation Inclusions

```java
/**
 * Write a description of class Start here.
 *
 * @author Mark Creado
 * @version version 1.0.0 21 June 2019
 */
public class StartGame
{
    //Maximum value supported
    private static final int maxvalue = 10;
    private int valueX;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        valueX = 0; // initialise instance variables
    }
}
```

```java
/*
 * Write a description of class Start here.
 *
 *@author (your name)
 *@version (a version number or a date)

public class StartGame
{
    //Maximum value supported
    private static final int maxvalue = 10;
    private int valueX;

    /**
     * Constructor for objects of class Start
     */
    public Start()
    {
        valueX = 0; // initialise instance variables
    }
}
```

- Document each method in a similar manner to a class, following the template given by BlueJ.

Correct Documentation Inclusions	Incorrect Documentation Inclusions

```java
/**
 * Constructor for objects of class StartGame
 */
public StartGame()
{
    valueX = 0; // initialise instance variables
}

/**
 * Method to get the person's name
 *
 * @param  newValue   an integer input to pass the id number of the person
 * @return            a single string to return the person name
 */
public String getPersonName(int newValue)
{
    int id = getId(newValue);
    Person personOne = new Person(id);
    return personOne.getName();
}

/**
 * Method to calculate the sum of two values passed
 *
 * @param  valueOne   an integer input to calculate the sum of
 * @param  valueTwo   an integer input to calculate the sum of
 * @return            the sum of valueOne and valueTwo
 */
public void calculateSum(int valueOne, int valueTwo)
{
    int id = getId(y);
    Person per = new Person(id);
    per.getName();
}
```

```java
/**
 * Constructor for objects of class StartGame
 */
public StartGame()
{
    valueX = 0; // initialise instance variables
}

/**
 * Method to get the person's name
 *
 */
public String getPersonName(int newValue)
{
    int id = getId(newValue);
    Person personOne = new Person(id);
    return personOne.getName();
}

/**
 * Method to calculate the sum of two values passed
 *
 */
public void calculateSum(int valueOne, int valueTwo)
{
    int id = getId(y);
    Person per = new Person(id);
    per.getName();
}
```