# Weka Grain

## Task List

Version 1.1

Following is the list of various tasks,so grab one and start working!

## ✔ *Task 1*

*enriching a data type.*

#File to be altered : **RandomDataProvider.java**

#example
```
public String getRandomComputerLanguage()
	{
		String
lang[]={"C","C+","C#","Java","Python","php","asm","VHDL","Shell","JavaScript","ASP
"};
		Random r=new Random();
		int index=(r.nextInt(10000))%lang.length;
		return lang[index];
	}
```

More languages can be added to the enumeration lang[] (highlighted in bold),ruby,fortran,smalltalk etc.

✔ Task 2

## *Adding Proper JavaDoc Comments*

#File to be altered RandomDataProvider.java

#example

```
/**
 * This is just a simple example to show how javadoc comments can be formatted .
 *
 * @param str
 *           The input string
 * @return The resulting string
 */
```

for example ,the comment to be added to getRandomGroceryItem() may look like
```
/**
 * Returns a grocery item
 *
 * @param str
 *            none
 * @return a string ( a random grocery item)
 */


public String getRandomGroceryItem()
      {
      String 
grocery[]={"Milk","Bread","Honey","Butter","Biscuits","Nuts","Pie","Cake","Juice",
"Eggs","Fruit","IceCream","Coffee","Tea"};
      Random r=new Random();
      int index=(r.nextInt(10000))%grocery.length;
      return grocery[index];

      }
```

# ✔Task 3

*ADDING A NEW DATA TYPE*

This is also an easy process but involvoes modifying code at 2-3 places.
#File to be altered : RandomStringProvider.java,GrainStringParser.java

#example:
We will shoe how to add a data type by adding a datatype called favourite IDE.
First of all we need to equip our random data provider with the ability to provide This datatype i.e. ,we should be able to instruct RandomStringProvider to supply a random IDE name.
A)Defining the datatype
1)Let us first define the data,a string of integers.

```
String IDEName[]={“Eclipse”,”NetBeans”,”Visual Studio”,”DevC++”};
```

2)having defined the data type we need to create a function who picks one random IDE name from the list and returns it to the caller.

The function prototype is as follows:

```
      String getRandomIDEName();
```

Note the function name,get<datatypename>.Please adhere to this notation.

3)Now we know how the function looks like,let us now add the logic.we basically generate a random number and then use this as an index to the array.the string thus found is returned.
The code looks as follows:

```
      public String getRandomIDEName()
      {
String IDEName[]={“Eclipse”,”NetBeans”,”Visual Studio”,”DevC++”};

      Random r=new Random();
      int index=(r.nextInt(10000))%IDEName.length; //get a random number
      return IDEName[index]; //return a random IDEName
      }
```

Just to remind you,we have to add this function to RandomDataProvider.java

So we are done with the data type,now we have to couple it with the code.

B) Adding the datatype to the list of datatypes.

1)Adding the data type to the list of datatypes (displayed on the left with orange background)
-open MainWindow.java
-find String infoStr
-append your datatype name to the list

-initially the list looked like:

```
String infoStr="Data Types Available\n1. name : Name  \n2. int_id : Integer ID" +
"\n3. dept: Department\n4. yn : Yes /No\n5. tf : True / False\n6. gender :" +"
Gender\n7. ph : Mobile Phone Number\n8. clang : Computer Language\n9. grocery :
Grocery Item\n10. state : Indian State and UT" +"\n11. weather:Weather
Conditions\n12. marks : Marks (out of 100) ";
```

-now we'll add our data type ide_name

```
String infoStr="Data Types Available\n1. name : Name  \n2. int_id : Integer ID" +
"\n3. dept: Department\n4. yn : Yes /No\n5. tf : True / False\n6. gender :" +"
Gender\n7. ph : Mobile Phone Number\n8. clang : Computer Language\n9. grocery :
Grocery Item\n10. state : Indian State and UT" +"\n11. weather:Weather
Conditions\n12. marks : Marks (out of 100)\n13. ide_name:Name Of an IDE ";
```

the format is as follows:
\n<number>.datatypename:short description

ok ,now we'll have to extend our parser ,GrainStringParser.java to add support for our new datatype.

C)Now we'll add the support to parse the datatype,ide_name
1.open GrainStringParser.java
2.find a list of “private static final int definition” and append your datatype at the end.obviously,you can add any name to describe the datatype.

```
private static final int
      ID=1,
      DEPT=2,
      YES_NO=3,
      TRUE_FALSE=4,
      GENDER=5,PHONE=6,
      NAME=7,
      COMPLANG=8,
      GROCERY=9,
      STATE=10,
      MARKS=11,
      WEATHER=12,
      IDENAME=13; <--added
```

3.next we will modify parseGrainString() function.
This function basically records with each attribute the data type the datatype

associated with it .
We basically add the code to assign this information.
For example :

```
else if(nameType[1].equalsIgnoreCase("weather"))
                  {
                        attribInfo[i]=WEATHER;
                  }
```

assigns the attribute number 4 to be of data type WEATHER
we add similar code for our own IDEName

```
else if(nameType[1].equalsIgnoreCase("ide_name"))
                  {
                        attribInfo[i]=IDENAME;
                  }
```

4)Next we extend arffWriter() and writeCSVData().
As the name suggests ,they write the data in the required format.
In both these functions,we have to add the case IDENAME to handle the data generation for this data type.

-Extending arffWriter()

```
case IDENAME:
      name="IDENAME_"+i.toString();
      type="{\"Eclipse\",\"NetBeans\",\"Visual Studio\",\"DevC++\"}";
                  / \
                   |
                   |
      Same as the datatype definition except that we have added quotes and \
break;
```

-Extending  writeCSVData(PrintWriter pw)

```
case DEPT:
attribVal=rdp.getRandomIDEName();
break;
```

Congrats,you have just added a new data type :).

## ✔Task 4

***Adding the readme file.***

If you feel that you know how WekaGrain works,write a readme for us explaining how to use the software.