Proposed Rough Edges of Interface-a Design Pattern

Anand Kr. Shukla
Research Scholar
Invertis University,
Bareilly

M. Zubair Khan
Deptt. of CS
DAS, Taibah University
Madinah
Saudi Arabia

Jagdish Rai
Professor
Invertis University,
Bareilly

ABSTRACT
Interface of java used as a design pattern for object oriented software development mostly used with inheritance, in java multiple inheritance is possible only through the interface this approach is widely used with modern software development approach apart from that there is also a rough side of java's Interface which discussed in this paper, interface doesn't allow the any kind of definition under it, but there is also some other concept like parent and child class concept if such concept apply with the interface then the theory of interface has been changed in this paper a practical approach has been used for this research problem.

Keywords
Designing pattern, Interface, rough edges.

1. INTRODUCTION
Design patterns are a way of implementing a common solution to a common problem in object-oriented software. Many informal catalogues exist, explaining how they are used and implemented. However, there is not any standardized way for defining about the consistence of design pattern for all the informal descriptions of design patterns

[1][2][3], java is consider as among the best suitable Object-Oriented Programming language which provides specially Multiple inheritance through the Interface, interface in java along with inheritance, multiple inheritance used commonly and mostly for design pattern and structure or software prototype, Interface is basically a container which contains only abstract methods which are the solution for recurring software design problems[6][7],there are many design patterns has been discovered and published, there are some design patterns can be directly supported by some programming languages, among them multiple inheritance gives a facilities to inherit data and code of multiple class (more than one class) in a derived class, multiple inheritance some time consider found matter of controversy and indiscipline because of its nature, there are some problem or dark side of this interface are like –problem of name clashes[16][17], complexity so some programming language like Java avoids multiple inheritance for becoming software architecture cleaner[2][6] and more simple, but sometimes there are many software development stages in which a software developer really need multiple inheritance, for this stages java gives Interface which used for providing the facility of Multiple inheritance and also provide a structural design for software development , but still there is some issues arrives in the interface, which will be discuses in this research paper, this will not wrong if said the loose poll of interface in java.

2. BACKGROUND
There are several other projects which has been focused on the design pattern they have created them from an empty class, improving existing design code and applying them into a design pattern. Meijers' Fragment Tool [5] which provides a mechanism from which a design pattern can be represented and also they could modified through a development tool. The main focus was to facilitate the use of design patterns for design and implementation, in-spite of focusing on the lower-level classes, methods and fields, in Object Oriented programming language like Java, Patterns also can be represented as fragments (inheritance or templates), which are associated with classes. Now developers has to decide that which fragment has to be chosen for better result keeping with the roles of other classes/methods/fields, pattern should also be decided on the basis of re-patterning, because it is possible that the pattern require change at any time.[4][5]

Budinsky et al. in (1996) developed a computer based tool[11], which automates the implementation of design patterns. For this tool a user can take specific input for the particular application for any given pattern from which the tool generates the entire pattern prescribed code automatically [11].

Krueger [12] introduces the following taxonomy to classify some different-different reusable approaches like software-component, architectures, schemas, application generators, and transformation systems, schemas of any software and architectures area for a reusable Software are the part of an design pattern in Object Oriented approaches. Reusing of a abstract schemas (abstraction), such abstractions are represented formally so they can be access automatically,the Paris system [13] is representative of schema technology. Schemas are lower than the Design patterns, In addition, design patterns cannot be instantiated directly because design patterns are not formal descriptions [12][13]

3. INTERFACE FOR DESIGN PATTERN
Interface in java used under the design pattern (Inheritance) for object oriented software development in java, multiple inheritance is only possible in java through Interface in which multiple group (Super class-which can creates by Interface) can be implemented together in the single class(Derived class) like fig 1,

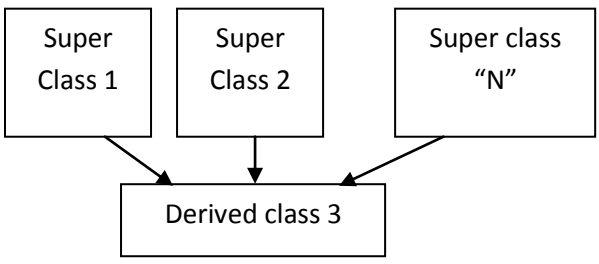

Fig-1.Multiple Inheritances as Design pattern (Only Possible through Interface in Java)

Interface is the collection of abstract methods only which can be creates by the interface keyword[13[14],

```
interface MyInterface
{
void message(); //abstract metods.
}
```

Fig- 2.declaring an interface

abstract methods are those methods which have only declaration (method prototype) include methods signature like behavior , Access modifier, return type and identifier only they haven't their definition in the interface[15],

```
public void Area(); // declaration of method
such types of methods can contain by interface.
interface Rectangle
{
public void Area();
}
```

Fig 3. Types and signature of the methodsupported by interface

Methods cannot be define inside the interface, this will just a designing structure for the software which is going to be developed, before performing coding need to create a prototype of that ,In Java method prototype cannot be created directly[13][9], because in Java method also have an parent body i.e classes and interfaces, in the class Java does not allow method declaration, and for software design need of method prototype then Java provide the concept of Interfaces, but java says that Interface can only be used for containing method prototype only it can't be define in the interface,

If programmers define the methods in the interface as show in the fig number 2.

```
interface CantDefine
{
void msg()
{
System.out.println("can not define the interface");
}
void msg2();
}
class Use implements CantDefine
{
public void msg2()
{
System.out.println("this is ok");
}}
class call
{
public static voidmain(String anand[])
{
Use u1=new Use();
u1.msg();
u1.msg1();
}}
```

Fig4.defining the interface

During the compilation of this code , the compiler generates errors –that Interface does not allow definition of the methods.

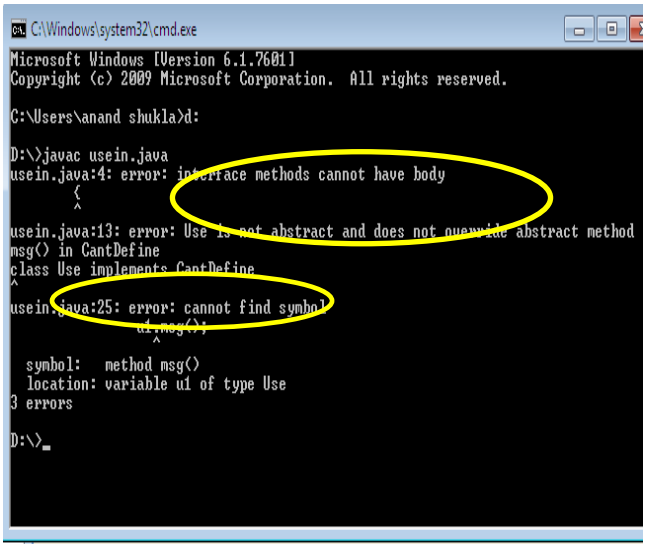

Fig 5.compilation the program of fig-04

The definition of the abstract methods can be give into the derived class which inherit the interface by implementing (by implements keyword) and then programmers can define the methods of interfaces, like below.

```
interface MyInterface
{
void message();
}
```

Fig-6. Interface which is going to be define in class

There is a need of a class in which programmer can implements the interface.

```
class UseRect implements MyInterface
{
public void message()
{
//Definition of the method of interface.
}
}
```

Fig 7. Creating and implementing the interface structure of implementation of an interface into a class

There is a class UseRect which inherits the interface by implements keyword, now is ready for defining the methods of the Interface as fig-08,

```
interface MyInterface
{
void message();
}
class UseInterface implements MyInterface
{
public void message()
{
System.out.println("definition of the method of a interface");
}
}
class call
{
public static voidmain(String anand[])
{
UseInterface ui=new UseInterface();
ui.message();
} }
```

Fig 8. implementation of an interface into a class

The class which defines the methods of interfaces are said to the derived class because first it inherit the interface then said to be derived class, now it have the rights to give the definition of the abstract methods of the interface

This is the way by which programmers design the structure for software in java.

4. ROUGH EDGES OF INTERFACE

Interface is really works very well for providing software designing structure, but also have a major Rough side of this wonderful concept of Java, java says that only method declaration is allowed under the interface no definition is allowed , but in practical this is possible that definition can be given under the interface let see how this is possible firstly there is need to understand the concept of parent and child class in java.

5. PARENTS AND CHILD CLASS APPROACH

Java provide the Parents and child class approach, in this any class can also be define under any other class , so the upper class which consist another class known as Parent class and the class [7][6] which reside under the parent class known as child class, for more understandable let's see an example .

```
class Parents
{
//Definition of parent class
        class child
        {
        //Definition of child class
        }//child closed
}//parent closed
```

Fig 9.Syntax of parent and child class

```
class Parents
{
    void msg1()
        {
System.out.println("I am parent");
        }
        class child
        {
            void msg1()
             {
System.out.println("I am child");
              }
} //Child closed
} //Parent closed
```

Fig 10.parent class example

Such arrangement of classes known as parent child class concept, this arrangement will compiled successfully (if no syntax error) by this command on dos.

Compiled by :\> javac nameofprog.java and then run by

:\> java Parents$child command.

This is possible because java creates separate-separate bytes code(class file) for each and every classes, the name of bytes code (class file) of parent class of above example will be Parent.class and name of child class will be as Parent$child.class as shown in below figure-11, [15][16]

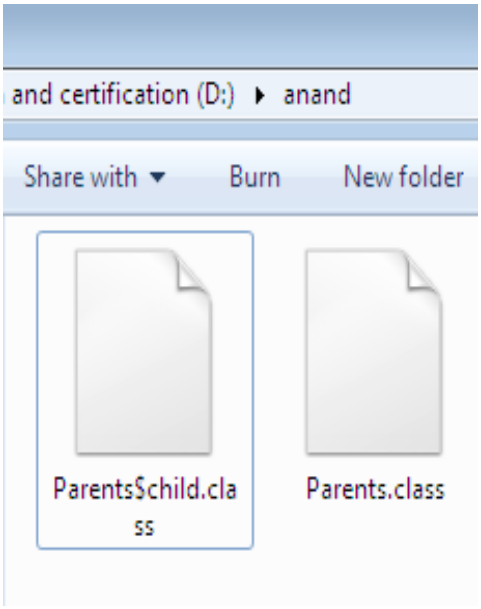

Fig 11. View of parent and child class in windows

In the case of interface , when programmers compile the interface , java's compiler also creates Byte code of interface in the form of class , and in above code this is clear that a class can contain another class . so if programmers assume interface as a class then programmers can apply the parent and child class concept , here we have to assume interface as parent class then programmers can write a child class into interface(parent class). [5][6]

6. RESULTS

```
interface  Identifier  //parent class
{
        class identifier  //child class
        {
        Methods()
        {
        //Definition of methods
        }
  } //Child closed
 } //parent closed
```

Fig 12. Structure of merging child class in to interface.

If programmers merge the concept of Parent child class into interface, without worrying that interface does-not allow the definition, like following.

```
Interface BreakingInterface
{
Class Breaks
{
Public static void main(String anand[])
{
System.out.println("Interface can also be define");
}}}
```

Fig 13. Defining the interface by parent and child concept.

Here given the definition into the interface, which are strictly denied by JAVA, but when programmers gave the definition by creating child class then result is changed, when compile by the java compiler as in figure -14, then

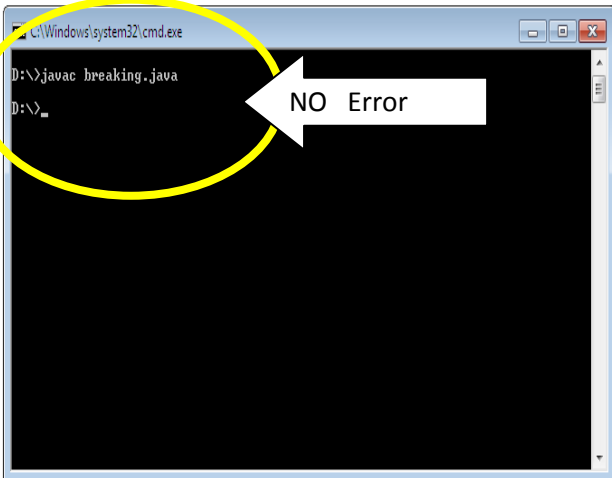

Fig 14.Compilation of fig-13's code.

there is no error occurs it means this code is being successful compile and converted into Byte codes, one Byte code for parent class (interface) and another byte code for child class, which breaks the concept that interface can't be define , now programmers can say that interface can be define, this can be run simply as parent child concept has ran as in figure 8.

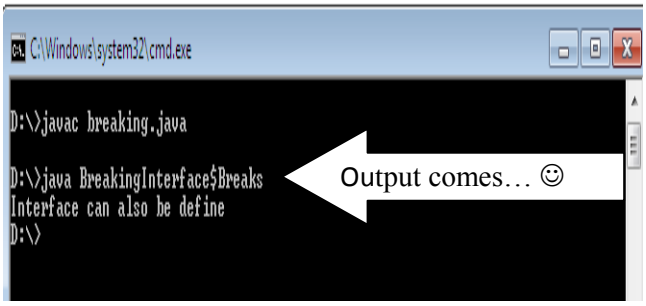

Fig 15.excecution of figure -13.

7. CONCLUSIONS

In java when programmers compile the interface, java Complier creates a class file (Byte Code) of interface means java Compiler treats interface as Class

and java already provided the concept of Inner class, so when programmers write a class into the interface then interface

behaves like an parents class which is same as a Parent-Child class concept, and compiler treated interface as an class, thus way compiler doesn't provide any error and creates the byte code which can be run by parent-child method using the parent&child command on the promt.

Dir:/> java Parent&Child

D:\> java BreakingInterface&Breaks

So the with this programmers can prove and say that programmers can break the limitation of one concept of Java by another concept of Java.

"If java prevent interface to convert into class file after compilation then this rough edge can be removed from interface , java has to convert interface into byte code but not in the form of class otherwise this will remain same ".

8. REFERENCES

[1] Wesley, 1995. ISBN 0-201-63361-2.*Summary of Changes to DoD-STD-2167A and DoD-STD-7935A resulting in MIL-STD-SDD*, Executive Summary, p. 1, December 1992.

[2] O. Agesen, S. Freund, and J. Mitchell, "Adding Type Parameterization to the Java Language", OOPSLA 1997, 49-65.

[3] G. Bracha and W. Cook, "Mixin-Based Inheritance", ECOOP/OOPSLA 90, 303-311.

[4] [BMRSS96] Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M.: Pattern-oriented Software Architecture: A System of Patterns. Wiley 1996. [CaW98] Campione M., Walrath K.: The Java Tutorial, 2nd edition, Addison-Wesley, 1998.

[5] [GHJV95] Gamma E., Helm R., Johnson R., Vlissides J.: Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley 1995.

[6] BUDINSKY, F., M. FINNIE, J. VLISSIDES and P. YU(1996) Automatic code generation from design patterns, IBM Systems Journal, 35 (2), 151–171.

[7] Charles W. Krueger. Software reuse. ACM Computing Surveys, 24(2), June 1992.

[8] S. Katz, C.A. Richter, and K.-S. The. Paris: A system for reusing partially interpreted schemas. In Proc. of the Ninth International Conference on Software Engineering. 1987

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides.Design Patterns: Elements of reusable object-orientedsoftware. Professional Computing Series. Addison

[10] E. Gamma, R. Helm, R. Johnson, and J. Vlissides.Design Patterns: Elements of reusable object-orientedsoftware. Professional Computing Series. AddisonWesley, 1995. ISBN 0-201-63361-2.

[11] Proc. of the Ninth International Conference on Software Engineering, 1987.

[12] Twin – A Design Pattern for Modeling Multiple Inheritance Hanspeter Mössenböck

[13] KRAMER, C. and L. PRECHELT (1996), Design Recovery by Automated search for structural design patterns in object oriented software, Proceedings of the Third Working Conference on Reverse Engineering, New York: IEEE, 208–215.

[14] [jacob99a] I Jacobson, Booch G. and Rumbaugh J., "The Unified Software Development Process." Addison-Wesley, Reading, MA, 1999.

[15] Kung, D. C., H. Bhambhani, R. Shah, and G. Pancholi. 2003, An expert system for suggesting design patterns: a methodology and a prototype. In Software Engineering With Computational Intelligence, ed. T. M. Khoshgoftaar. Kluwer Int.

[16] J. Coplien. Pattern Languages of Program Design 1.Pattern Languages of Program Design. Addison Wesley, June 1995

[17] M. Meijers. Tool support for object-oriented design patterns. Master's thesis, Utrecht University, 1996 INF-SCR-96-28.

[18] M. Fowler and K. Scott. UML Distilled.Addison-Wesley Professional, September 2003.