**Chapter 6**

TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product.

It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

The following are the Testing Methodologies:

1. Unit Testing.
2. Integration Testing.
3. Output Testing.
4. Validation Testing.

**6.1 Unit Testing**

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing paths are tested for the expected results. All error handling paths are also tested.

**6.2 Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

**6.2.1 Top-Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

**6.2.2 Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

**6.3 Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration.

**6.4 Validation Checking**

Validation checks are performed on the following fields.

**6.4.1Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

**6.4.2 Mobile Number Field:**

The mobile number field can contain only numbers from 0 to 9. An entry of any other character flashes an error messages.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

**6.5 Test Results for Sample DATA (Cab Requester & Provider)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **Commands** | **Data** | **Result of the selected operation** |
| EntryForm | Next | Destination, Pick Up point, Mobile number | Next screen with a list of drivers who have accepted |
| No entry in Destination or pick up OR less than 10 digits/or alphabets in the Mobile number | Alert screen indicating no input OR alert sound in case of error in entering mobile number |
| Restart | With Input | Starts from the Setup Module Again |
| Without Input | Starts from the Setup Module Again |
| Exit | With Input | Terminates the application |
| Without Input | Terminates the application |
| SelectForm | Next | Single choice Selected | Goes to the next screen for passcode and time |
| No choice OR multiple selected | Alert screen indication wrong selection |
| Restart | With Input | Starts from the Setup Module Again |
| Without Input | Starts from the Setup Module Again |
| Exit | With Input | Terminates the application |
| Without Input | Terminates the application |
| Finalize | Next | Passcode and Meeting Time (2 digits only) | Next screen indicating the finalization of the booking |
| Passcode/Meeting time < or > 4/2 digits OR alphabets | Alert screen indicating no input OR alert sound in case of error in entering mobile number |
| Try Again | With Input | Starts from the Setup Module Again |
| Without Input | Starts from the Setup Module Again |
| Finish | With Input | Terminates the application |
| Without Input | Terminates the application |

Table 6.1 Data testing for Requester Application

|  |  |  |  |
| --- | --- | --- | --- |
| Module | **Commands** | **Data** | **Result of the selected operation** |
| Setup | Ok | Single choice Selected | Goes to the next screen for displaying the list |
| No choice OR multiple selected | Alert screen indication wrong selection |
| Exit | With Input | Terminates the application |
| Without Input | Terminates the application |
| DisplayList | Next | Single choice Selected | Goes to the nest screen for displaying the list |
| No choice OR multiple selected | Alert screen indication wrong selection |
| Restart | With Input | Starts from the Setup Module Again |
| Without Input | Starts from the Setup Module Again |
| Exit | With Input | Terminates the application |
| Without Input | Terminates the application |
| AcknowledgeForm | Next | Yes Or No | Sends the message to the customer indicating the choice on the booking |
| No choice OR multiple selected | Alert screen indication wrong selection |
| Restart | With Input | Starts from the Setup Module Again |
| Without Input | Starts from the Setup Module Again |
| Exit | With Input | Terminates the application |
| Without Input | Terminates the application |
| Finalize | Try Again | With Input | Starts from the Setup Module Again |
| Without Input | Starts from the Setup Module Again |
| Finish | With Input | Terminates the application |
| Without Input | Terminates the application |

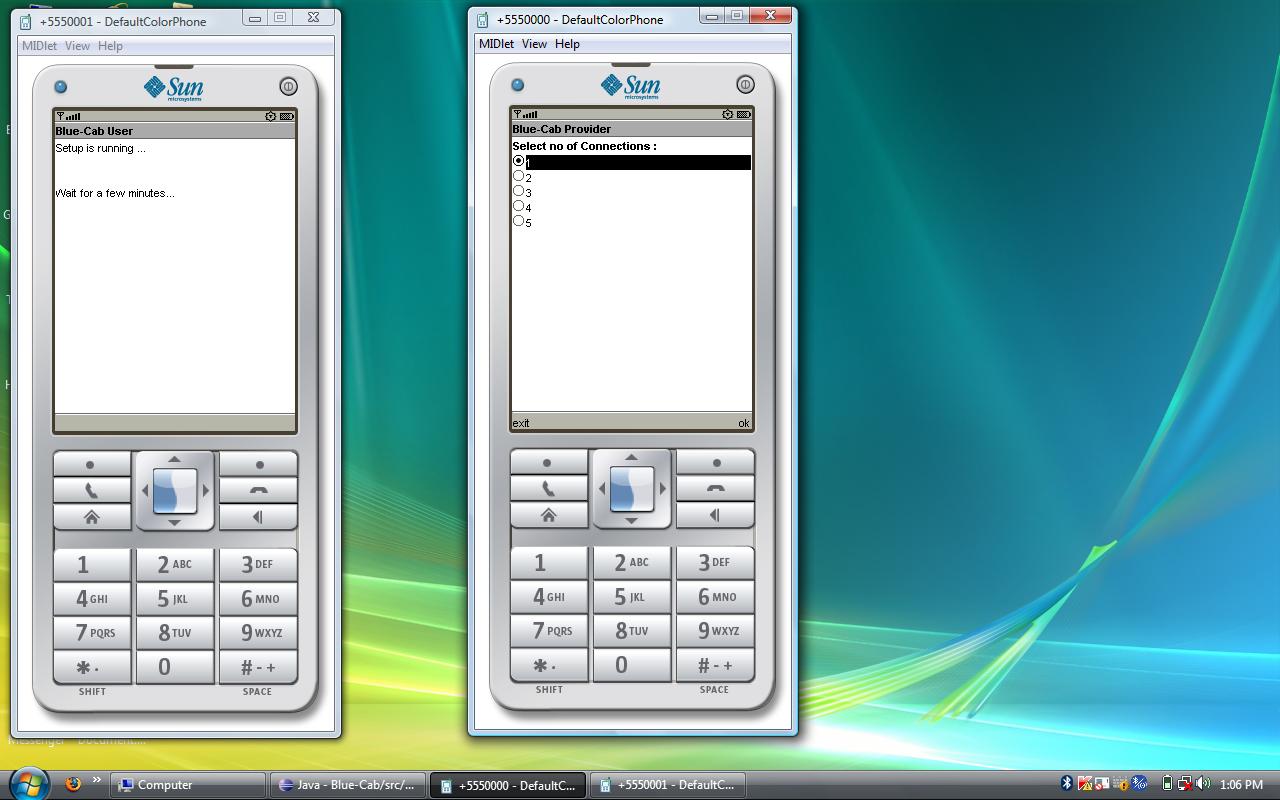
Table 6.2 Data testing for Requester Application

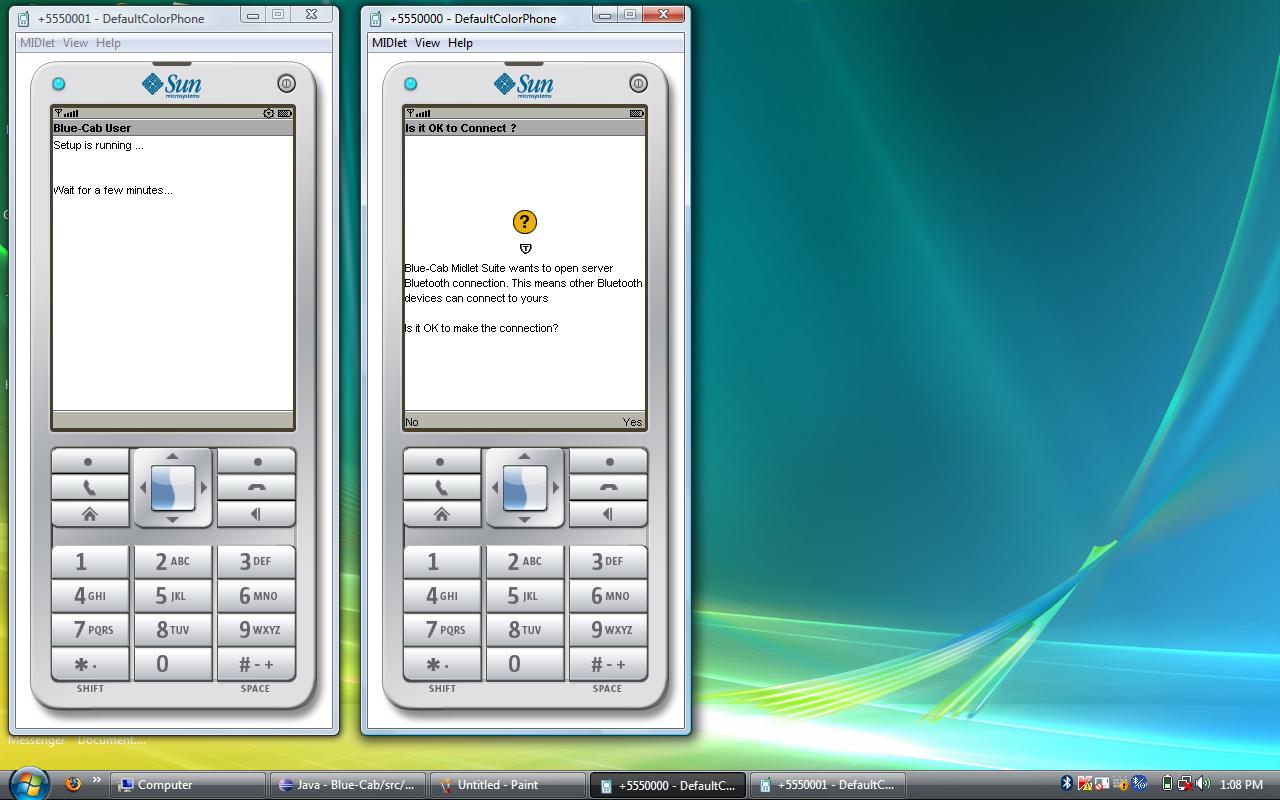
REMARKS:

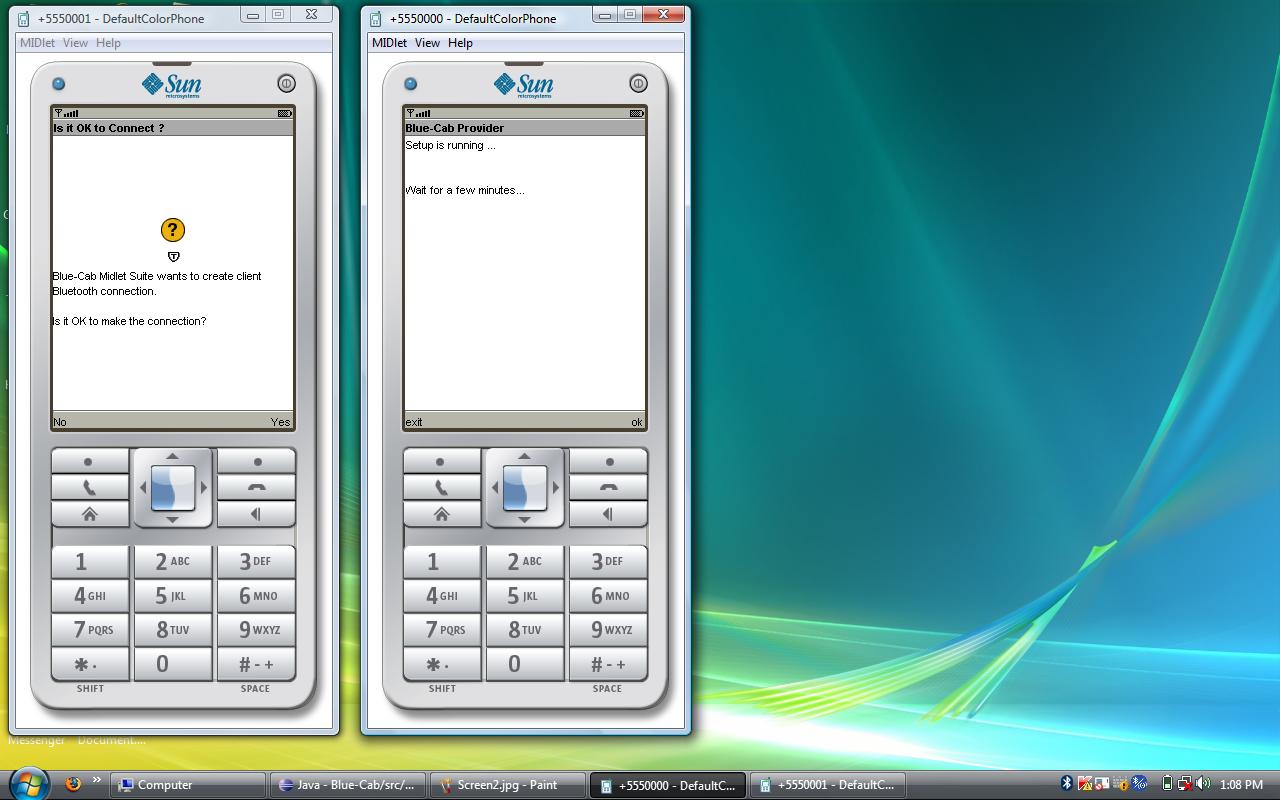
1. The application is command centric i.e. the choice of the user of the application drives the future actions or the course taken.
2. At any point of time if the user wishes to exit the application, it is made possible by providing every screen with Exit command.
3. At any point of time if the user wishes to Restart the application, it is made possible by providing every screen with Restart command.
4. Invalid entries are handled by the objects that are designed to obtain input (Textfields, Choicegroup) or sometimes manually by the user (checking for size).
5. In case of exceptions the system exits by catching the exceptions and displaying the stack trace.

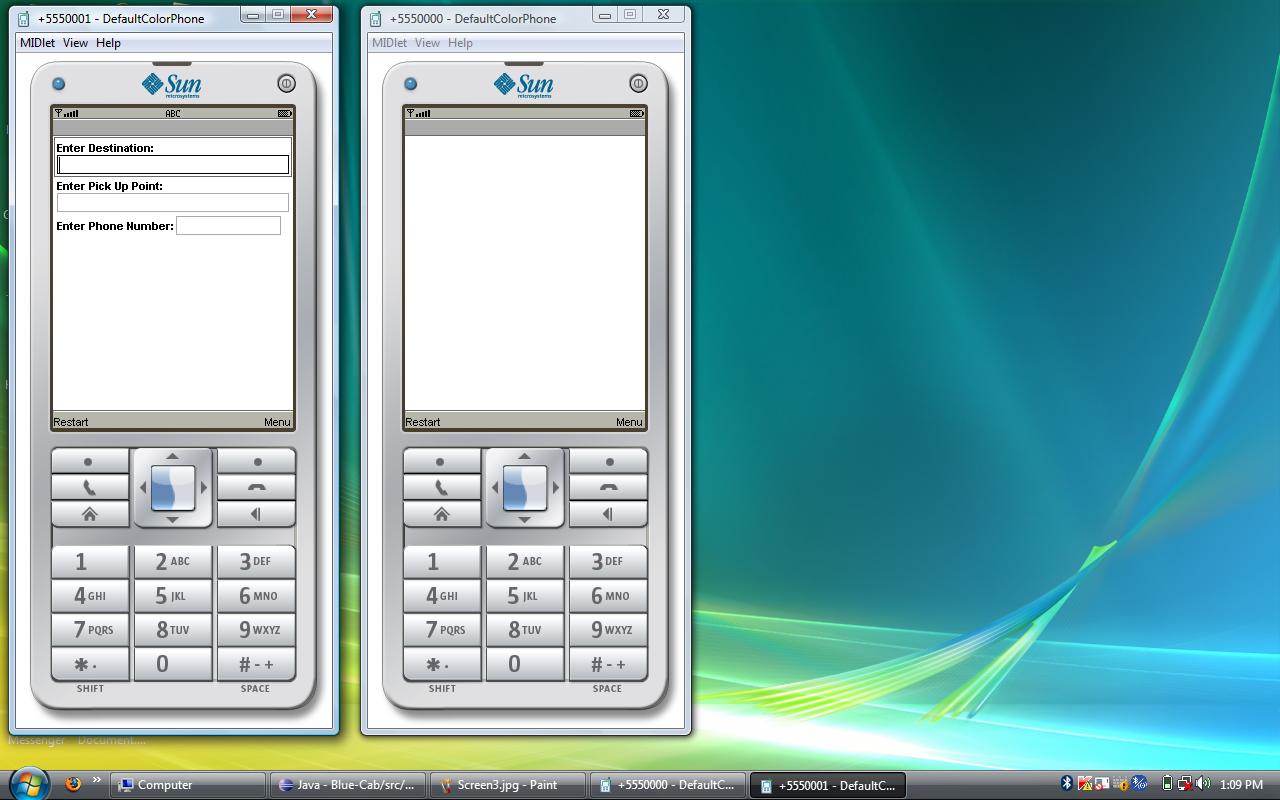
**CHAPTER 7**

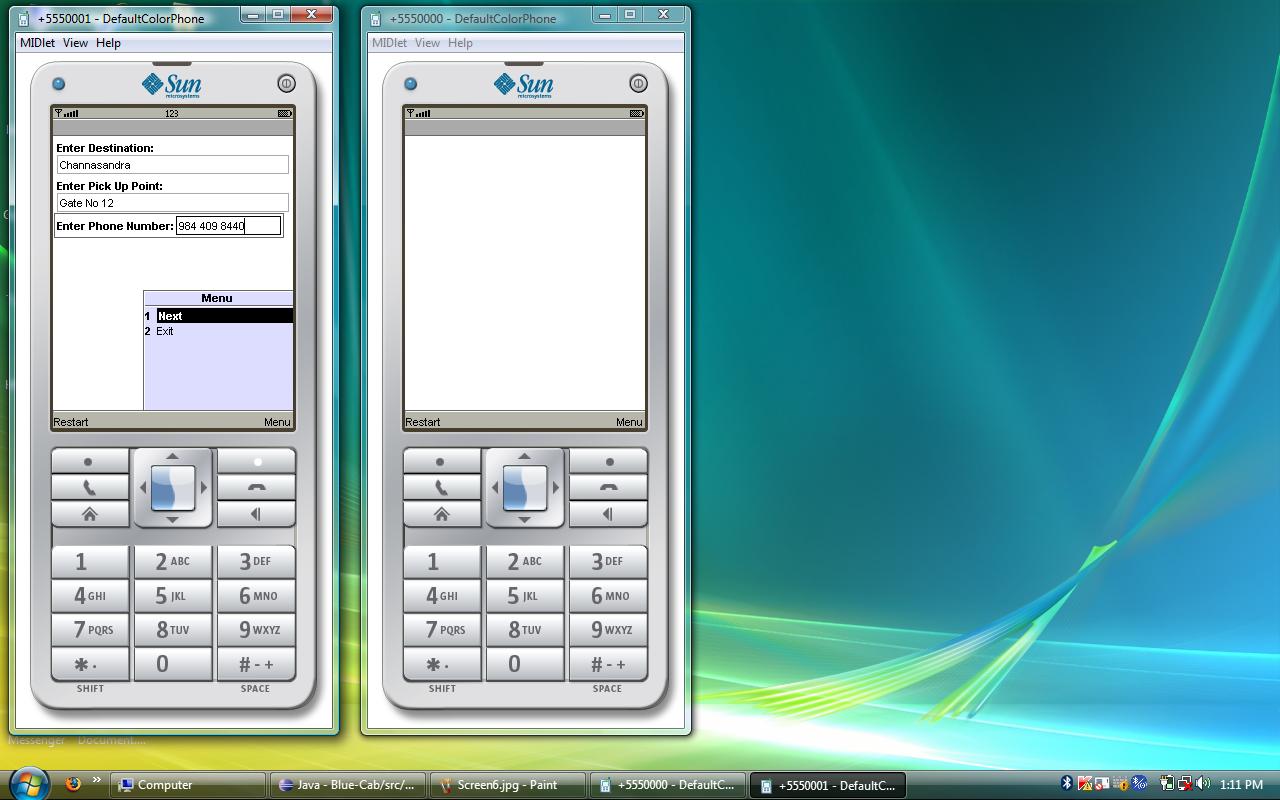
**SNAPSHOTS**

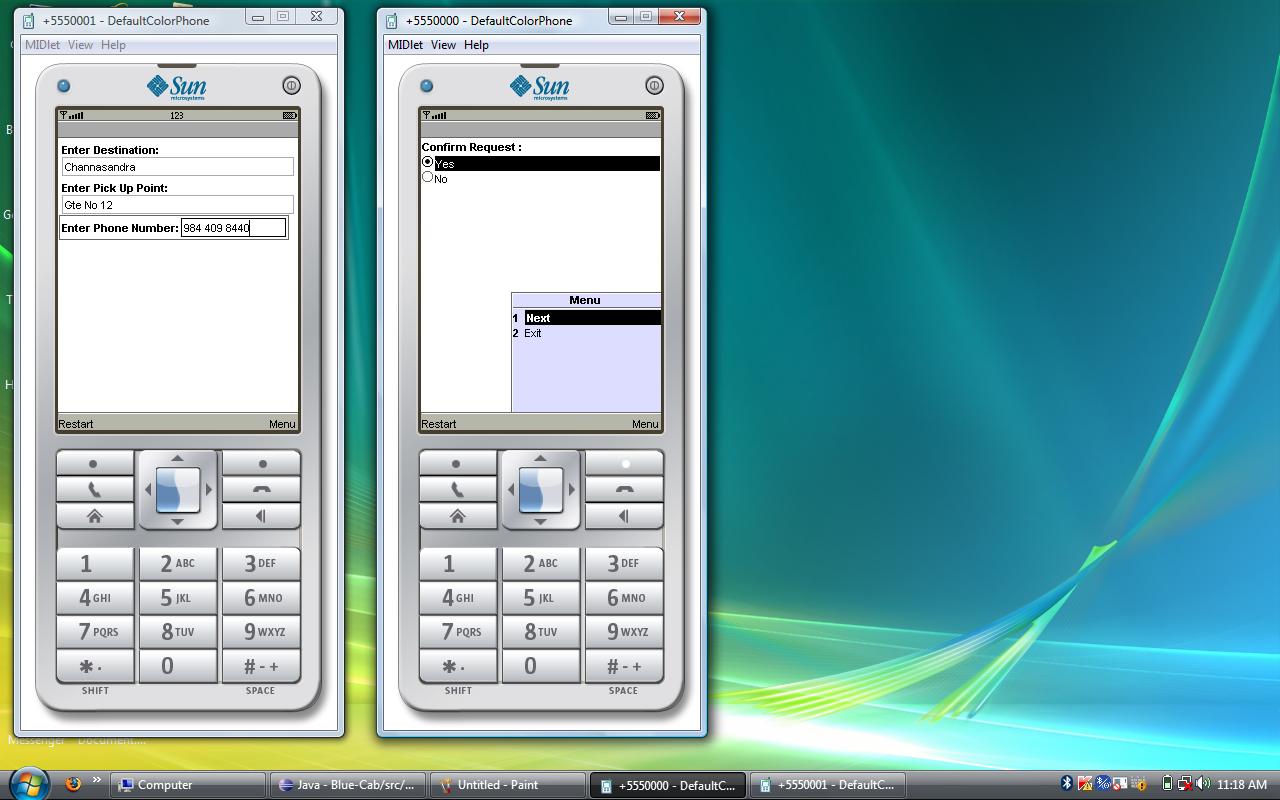


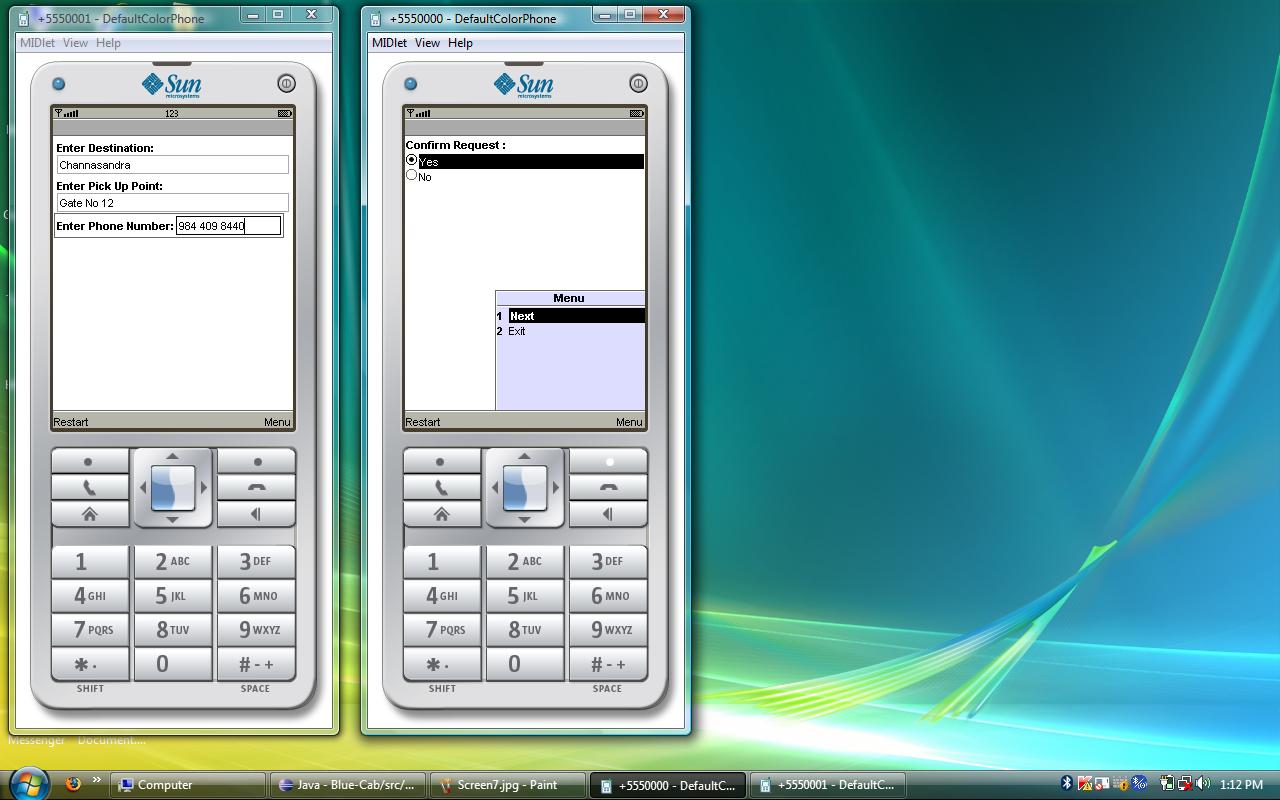


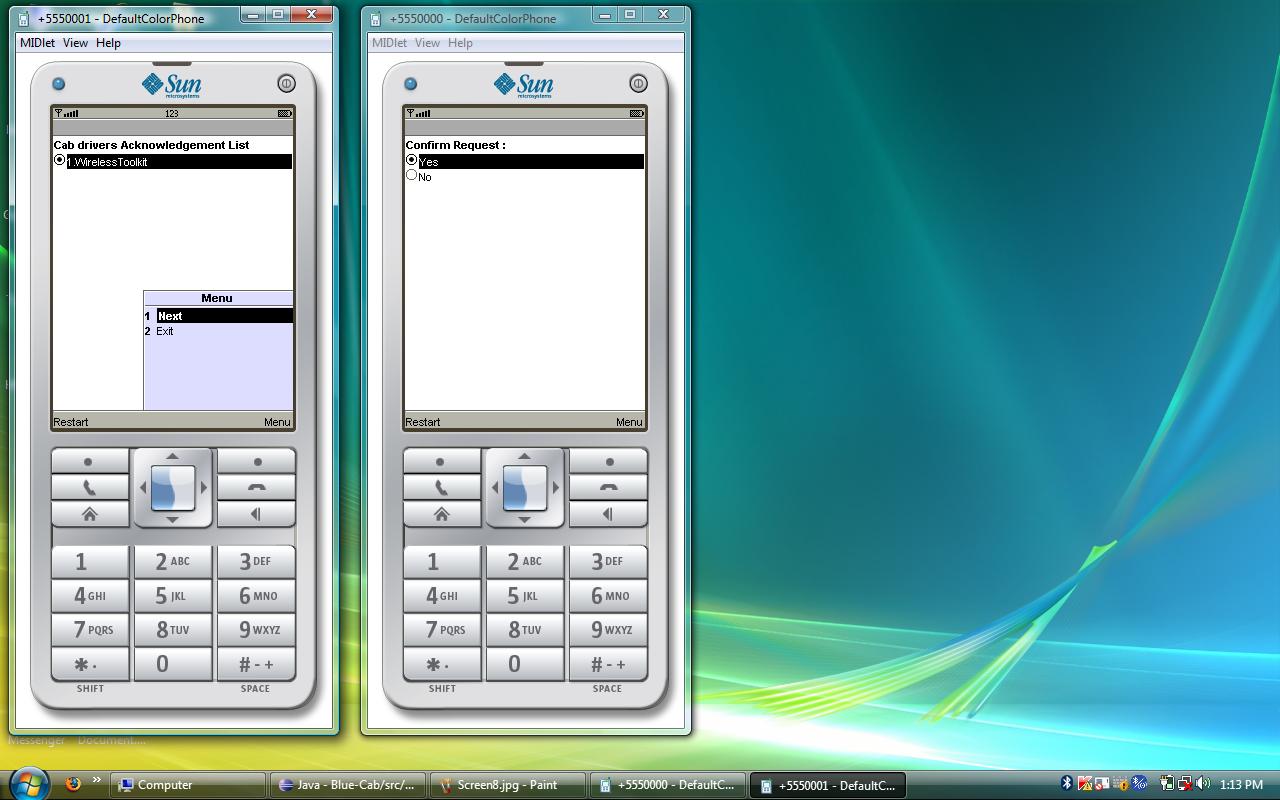


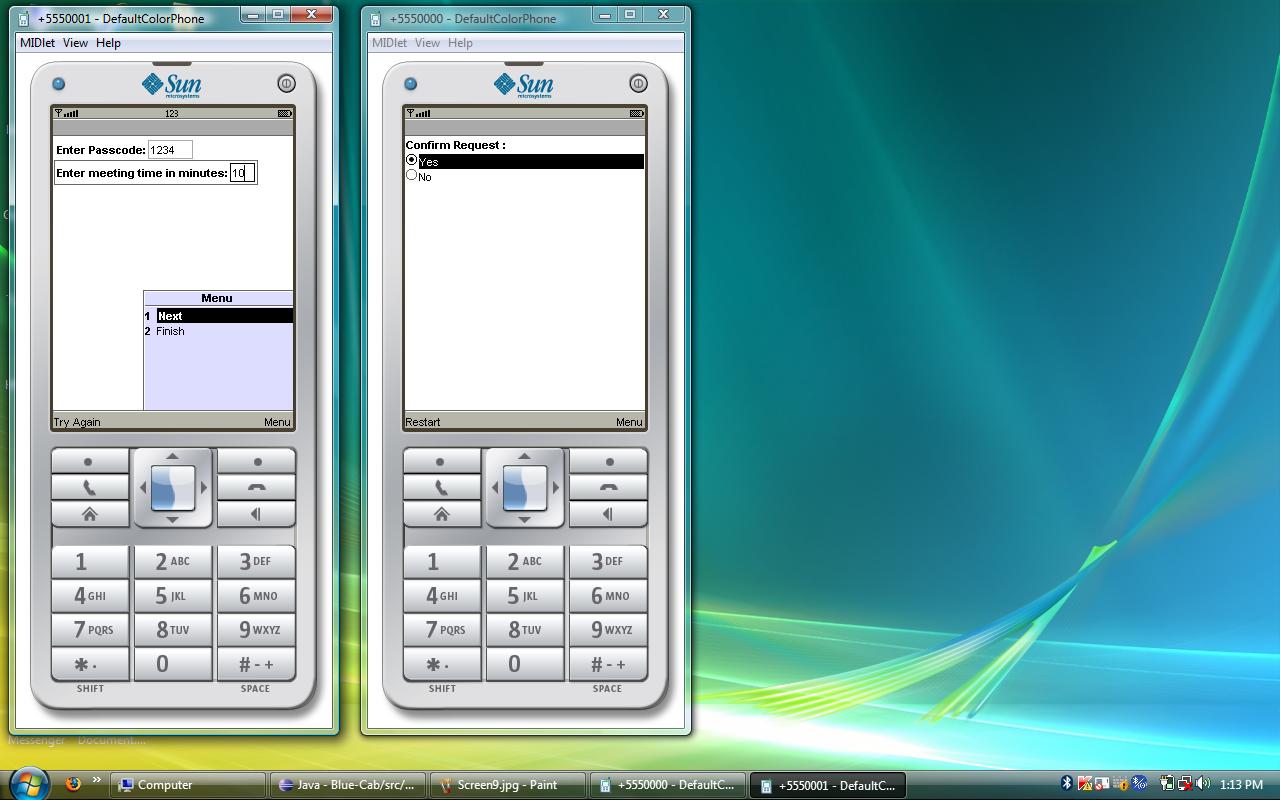


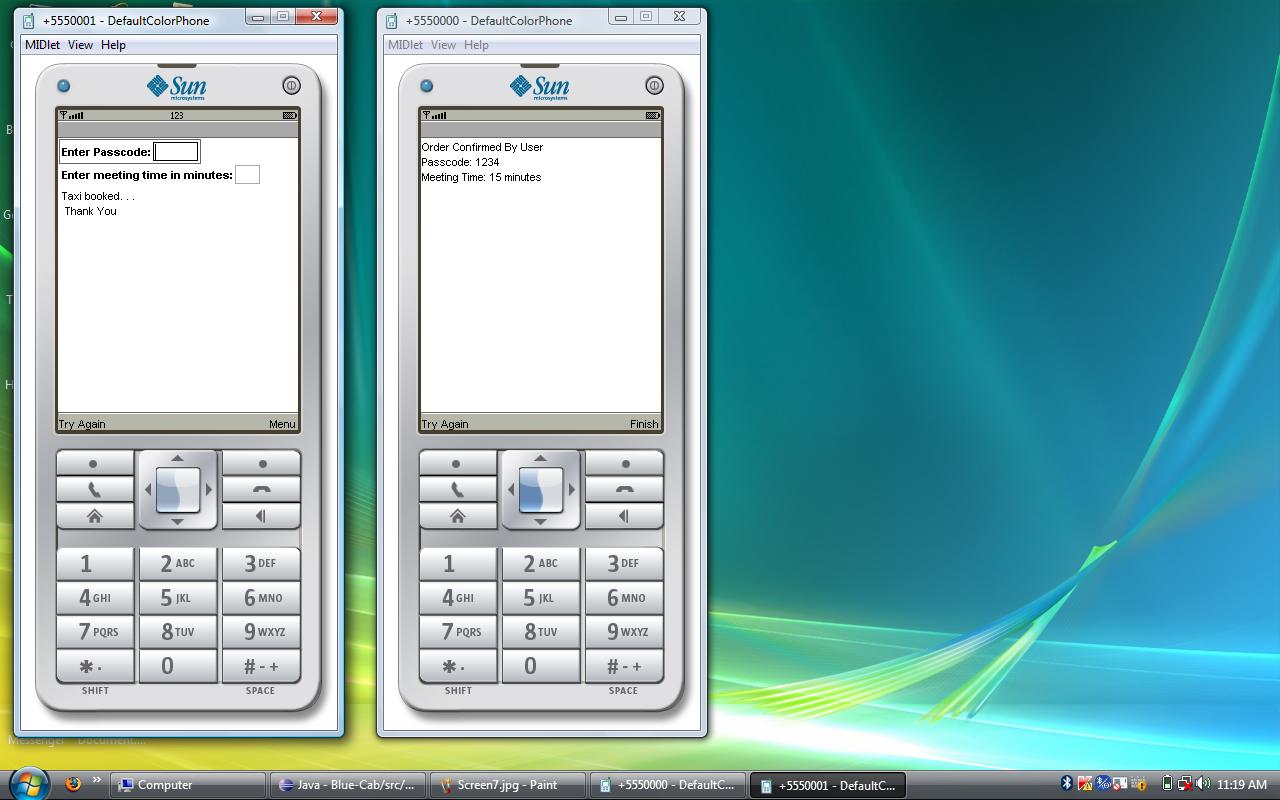












**CHAPTER 8**

CONCLUSION & FUTURE ENHANCEMENT

**8.1 CONCLUSION**

* Bluetooth used in most applications have only file transfer or audio transfer. With this project we intend to use this technology in a more interactive way providing a useful service without many hassles.
* Also the overheads (commission, user cost) and the waiting in queues involved in traditional existing systems can be eliminated.
* More importantly it provides an easy and direct access which is intuitive in nature for customers with almost no extra or hidden costs.
* This can be used as an advertisement tool as well.
* With this project we have been able to move a step forward in bringing technology to everyday life’s operations and making the inefficient jobs more efficient for the masses.

**8.2 FUTURE ENHANCEMENT**

* Achieving the difficult task of portability on all mobiles is the first priority and the range extension.
* Currently Wi-Fi is used in limited number of phones when compared to Bluetooth. So when Wi-Fi is embedded in most of the cell phones then Wi-Fi can be used to implement the same application.
* Adding more features such as connection to GPS system for a path to the destination and features local to some area (such as platform ticket reservation in Railway stations or pre-ordering of food in a crowded restaurant).
* Synchronizing it with the local PC for greater reach and control in areas with high rate of cab usage.
* Automating the entire system of booking a cab for a city.

**BIBLIOGRAPHY**

1. J2ME Bluetooth Programming -André N. Klingsheim
2. http://wiki.java.net/bin/view/Mobileandembedded/BluetoothResources
3. Developing Applications with the Java APIs for Bluetooth™ (JSR-82)-Sony Ericsson
4. Bluetooth Essentials for Programmers- Albert S. Huang and Larry Rudolph
5. http://developers.sun.com/mobility/
6. Symbian OS: Designing Bluetooth Applications In C++ - Nokia Corporation
7. Bluetooth for Javaby Bruce Hopkins and Ranjith Antony
8. http://www.palowireless.com/bluetooth/