# Programming Style

The programming language that has been chosen for this project is Java, as this is the language that everybody in the group is familiar with and thus it means that everybody can read, analyse and contribute to the programming section of the project, if need be.

Below is a class taken from the book “Objects First with Java: A Practical Introduction Using BlueJ” (2008) by David J. Barnes and Michael Kölling. The reason this has been used as an example of the programming style that we will use is that it is the style that the group members were taught to use.

|  |
| --- |
| /\*\*  \* TicketMachine models a ticket machine that issues  \* flat-fare tickets.  \* The price of a ticket is specified via the constructor.  \* Instances will check to ensure that a user only enters  \* sensible amounts of money, and will only print a ticket  \* if enough money has been input.  \*  \* @author David J. Barnes and Michael Kolling  \* @version 2008.03.30  \*/  public class TicketMachine  {  // The price of a ticket from this machine.  private int price;  // The amount of money entered by a customer so far.  private int balance;  // The total amount of money collected by this machine.  private int total;  /\*\*  \* Create a machine that issues tickets of the given price.  \*/  public TicketMachine(int ticketCost)  {  price = ticketCost;  balance = 0;  total = 0;  }  /\*\*  \* @Return The price of a ticket.  \*/  public int getPrice()  {  return price;  }  /\*\*  \* Return The amount of money already inserted for the  \* next ticket.  \*/  public int getBalance()  {  return balance;  }  /\*\*  \* Receive an amount of money in cents from a customer.  \* Check that the amount is sensible.  \*/  public void insertMoney(int amount)  {  if(amount > 0) {  balance = balance + amount;  }  else {  System.out.println("Use a positive amount: " +  amount);  }  }  /\*\*  \* Print a ticket if enough money has been inserted, and  \* reduce the current balance by the ticket price. Print  \* an error message if more money is required.  \*/  public void printTicket()  {  if(balance >= price) {  // Simulate the printing of a ticket.  System.out.println("##################");  System.out.println("# The BlueJ Line");  System.out.println("# Ticket");  System.out.println("# " + price + " cents.");  System.out.println("##################");  System.out.println();  // Update the total collected with the price.  total = total + price;  // Reduce the balance by the prince.  balance = balance - price;  }  else {  System.out.println("You must insert at least: " +  (price - balance) + " more cents.");    }  }  /\*\*  \* Return the money in the balance.  \* The balance is cleared.  \*/  public int refundBalance()  {  int amountToRefund;  amountToRefund = balance;  balance = 0;  return amountToRefund;  }  } |

To be more specific about the style of programming, here are some basic guidelines for the programming team;

* All code must be suitably commented and each class must come with an introduction to explain its purpose. The author/s and contributors to a class must be credited.
* The focus is not just on writing good programs, but also writing programs which are easily readable.
* Classes, attributes and methods must be given names which are suitable for their purpose.
* The title of the class is the only part of the actual program that is not indented.
* Methods must be properly indented i.e. all programming constructs must be indented and highlighted properly in order to maintain readability.
* A suitable amount of space must be left in between each method.
* All elements of a class must be labelled and explained in comments.
* Remember the importance of suitable character cases; attributes and methods written in lower case, with additional words beginning with a capital letter. Classes and constructors begin with upper case letters.