To start the darts project, A UML diagram aided to understand what classes, variables and functions were needed to be able to make the program efficient and to be able to include all three game modes within one program. This led to four classes being made which were, Player, DartBoard, Dart501 and GameMode.

The Player class held all the variables and functions relating to the players in the game which was separated into private and public, DartBoard was used to hold any functions and variables that were needed to play any of the dart game-modes. Dart501 was used to hold the functions and variables for the 501 darts game mode furthermore, Darts501 is a class that inherited variables and functions from DartBoard rather than having duplicate data being made. The GameMode class held all three types of gamemodes including, 301, 501 and 501 interactive version. Afterwards Pseudocode was created to plan out the program structure and where the functions from the UML diagram would be used to efficiently create the program. Once the pseudocode was done the program could be implemented.

The Artificial intelligence used for the bots in all the modes for darts include the bot throwing for the highest possible value until they reach zero and for 501, knowing when to throw for an odd number to ensure that they can finish on a double or a bullseye.

The interaction for the player allows them to choose who is going first in 301, the name and accuracy for both players, in 501 interactive the player uses a text-based interface to allow them to select what value they are wanting to throw for and what they would like to multiply it by, either single, double, or treble. It then uses the throwing functions to see if they hit their target or not based on their accuracy. The code makes use of ASCII to liven up the interface and clears the interface to make it easier for the player to read.

The object orientated programming used for term two has made keeping track of code easier and less stressful since any variables and functions related to the players is not stored in a big bundle of code in one .cpp file, instead it is stored within its own class that is isolated and easy to export for other programs as it is in its own file. It also makes the program more robust and secure since the program can only access the variables within the classes through setters and getters can be used to ensure variables can be used without being changed rather than being public for anyone to change. It also makes the code look tidy, neat, and readable for anyone else looking at the code, and it makes it easier to comment on the code and the main section will not have any functions in it as they will all be in the classes they belong to, so those functions can be reused in another class.

Finally object orientated programming allows inheritance of classes which means it is more efficient as there is no duplicates in the classes since it is inheriting functions and variables from another class, and it can overload the functions in the inherited class to have more than one purpose which increases efficiency further.