Term	Definition

Array	A data structure that can store multiple data items under the same name. These data items must all be of the same type.
	`private String[] toppings;`
	`public Pizza(){`
	`  toppings = new String[5];`
	`  toppings[0] = "cheese";`
	`}`
	Here, an array called 'toppings' is declared (of type string) and subsequently dimensioned with five spaces in which five separate strings can be stored. These spaces, called **elements** can be separately accessed in the way shown on the following line. The first element (element zero) is given the value cheese, and the remaining elements (elements one to four) are left empty.

ArrayList	A data structure that changes size as data items are added and removed from it.
	`private ArrayList<String> toppings;`
	`public Pizza(){`
	`  toppings = new ArrayList<String>();`
	`  toppings.add("cheese");`
	`}`
	Here, an ArrayList called 'toppings' is declared (of type string) and subsequently initialised. A string 'cheese' is added, and data are stored in an ArrayList in the order in which they are added.

Cohesion	A measure of the extent to which a single class models a single thing, and a single method performs a single task. High cohesion, in which these precepts are observed, makes software easier to maintain.

Constructor	A special-purpose method found within a class. When the constructor is called, one instance of that class is created. The name of a constructor always matches the name of its class
	`public class Race`
	`{`
	`  public Race() {`
	`  }`
	Here, we see the constructor within the Race class. Note that the constructor has no return type – not even 'void'

Coupling	A measure of the interconnectedness between classes. Loose coupling entails minimal interconnection. Tight coupling, which entails greater interconnection, presents problems in terms of maintainability.

Term	Definition

else	Used alongside 'if' to specify multiple paths through the code:
	`if (i < 5)`
	`{`
	`    a += 5;`
	`}`
	`else {`
	`    a += 4;`
	`}`
	`a += 3;`
	If 'i' is less than 5, the first block of code will execute (a += 5). If 'i' is not less than 5, the second block will execute (a += 4). In all cases, one of these two options, and only one of them, will occur. Also in all cases, the final line of code will execute (a += 3), since this is not inside the 'if' block or the 'else' block.

Field / Attribute	A named data structure within a class that stores information about an object. Each field consists of a visibility modifier, a type and a name:
	`private int result;`
	`private Race marathon;`
	Here are two fields. Both are private, meaning they can only be accessed from within the class in which they are declared. The field named 'result' is of type 'int', so it can store a single whole number. The field named 'marathon' is of type 'Race', which is a class.

final	A modifier found in field declarations that ensures that the field, once it has a value, keeps that value and never allows it to be changed. `private final double VAT = 0.2;` This field is now a **constant** – it cannot have its value changed. Convention dictates that names of constants are in all caps.

for	A type of loop for situations in which a section of code will repeat a predetermined number of times (rather than looping until a condition is met)
	`for (int i = 0; i < 5; i++) {`
	`}`
	The contents of this loop will run 5 times. The 'for' line itself is split into three parts:
	• `int i = 0` – this occurs once, before the first iteration, in this case declaring an integer variable called 'i' and setting it to zero.
	• `i < 5` – this states the condition under which the loop's contents will run; if 'i' is less than '5', the contents will be executed
	• `i++` - this specifies what happens between loops; each time an iteration ends, 'i' is incremented in this case.

Term	Definition

if	Specifies that a body of code will either be executed or not executed:
	`if (i < 5)`
	`{`
	`  a += 5;`
	`}`
	`a += 4;`
	If 'i' is less than 5, the instruction within the 'if' structure will be executed. Otherwise, it will not be executed. So here, 'a' is either increased by 5 or it is not. Under all circumstances, however, 'a' is increased by '4', since this instruction is not inside the 'if' structure.

Method	A named body of code within a class that performs some task. A Java method signature (the first line, which identifies the method), consists of four elements – a visibility modifier, a return type, a name and a parameter list:
	`public double getResult(int index) {`
	`  return result;`
	`}`
	This method is public (can be called from outside the), double (returns a date item of type double), named getResult, and requires a single parameter, which is named 'index' and is of type 'int'.

Parameter	An additional piece of data passed into a method in order for that method to be able to perform its job. The following method has no parameters:
	`public void refresh(){`
	`}`
	We can tell it has no parameters are the round brackets immediately after 'refresh' are empty. The brackets still need to be included, even if there are no parameters. The following method has two parameters:
	`public void refresh(int rate, String label){`
	`}`
	Each parameter comprises a name, followed by a type; multiple parameters for the same method are separated by a comma.

static	A modifier found in some method signatures and field declarations. The following is an example of a static field: `public static int score;` Even though this is declared in a class, it can be accessed and manipulated before an object of that class exists. Once objects are created using this class, no matter how many objects there are, they would all share access to this single field.

Term	Definition

while	A type of loop suitable for situations in which a section of code will repeat until a condition is met (rather than looping a predetermined number of times).
	`while (score < 100) {`
	`}`
	Any code within the curly brackets would execute repeatedly as long as 'score' is less than 100. If 'score' is always less than 100, the loop will execute indefinitely.