Boulder Valley School District
Excellence and Equity

Middle School Java Programming Curriculum Essentials Document

Boulder Valley School District
Department of CTEC
February 2013

Boulder Valley School District Technology – An Introduction to The Curriculum Essentials Document

Background

* This BVSD Curriculum Essentials Document incorporates the International Society for Technology in Education's (ISTE) National Educational Technology Standards for Students (NETS) and the integrated essentials from the Colorado Academic Standards for 21st Century Learning Skills.

The NETS for Students from ISTE do not delineate how courses should be created or taught. Each teacher must determine appropriate lesson planning. As technology rapidly evolves with new dynamic tools, there is no set of prescribed software, tools, or technologies that students and teachers may adopt to achieve these rigorous and robust standards. It is with experience, trust, and teacher consensus in ISTE that the Technology Teachers and Educational Technology Department at BVSD adopted these same NETS for our Boulder Valley students. The writing teams took the ISTE NETS for Students and carefully and thoughtfully divided them into courses for the creation of the 2011 BVSD Educational Technology Curriculum Essentials Documents (CED).

The ISTE 2007 Standards

The expectations in these documents are based on mastery of the topics at specific grade levels with the understanding that the standards, themes and big ideas reoccur throughout PK-12 at varying degrees of difficulty, requiring different levels of mastery. The Standards are:

1. Creativity and innovation

Students demonstrate creative thinking, construct knowledge, and develop innovative products and process using technology.
Students:
a. Apply existing knowledge to generate new ideas, products, and processes
b. Create original works as a means of personal or group expressions
c. Use models and simulations to explore complex systems and issues
d. Identify trends and forecast possibilities

2. Communication and Collaboration

Students use digital media and environments to communicate and work collaboratively to support individual learning and contribute to the learning of others.
Students:
a. Interact, collaborate, and publish with peers, experts, or others employing a variety of digital environments and media
b. Communicate information and ideas effectively to multiple audiences using a variety of media and formats
c. Develop cultural understanding and global awareness by engaging with learners of other cultures
d. Contribute to project teams to produce original works or solve problems

3. Research and Information Fluency

Students apply digital tools to gather, evaluate, and use information.
Students:
a. Plan strategies to guide inquiry
b. Locate, organize, analyze, evaluate, synthesize, and ethically use information from a variety of sources and media
c. Evaluate and select information sources and digital tools based on the appropriateness to specific tasks
d. Process data and report results

4. Critical Thinking, Problem Solving, and Decision Making
Students use critical thinking skills to plan and conduct research, manage projects, solve problems and make informed decisions using appropriate digital tools and resources.
Students:
a. Identify and define authentic problems and significant questions for investigation
b. Plan and manage activities to develop a solution or complete a project
c. Collect and analyze data to identify solutions and/or make informed decisions
d. Use multiple processes and diverse perspectives to explore alternative solutions

5. Digital Citizenship
Students understand human, cultural, and societal issues related to technology and practice legal and ethical behavior. Students:
a. Advocate and practice safe, legal, and responsible use of information and technology
b. Exhibit a positive attitude toward using technology that supports collaboration, learning, and productivity
c. Demonstrate personal responsibility for lifelong learning
d. Exhibit leadership for digital citizenship

6. Technology Operations and Concepts
Students demonstrate a sound understanding of technology concepts, systems, and operations. Students:
a. Understand and use technology systems
b. Select and use applications effectively and productively
c. Troubleshoot systems and applications
d. Transfer current knowledge to learning of new technologies

Components of the Curriculum Essentials Document
The CED for each grade level and course include the following:
An At-A-Glance page containing:
o approximately ten key skills or topics that students will master during the year
o the general big ideas of the grade/course
o the Standards of Technology Practices
o assessment tools allow teachers to continuously monitor student progress for planning and pacing needs
o Description of Technology at that level
The Grade Level Expectations (GLE) pages.
The Grade Level Glossary of Academic Terms lists all of the terms with which *teachers should be familiar and comfortable using during instruction. It is not a comprehensive list of vocabulary for student use*

Middle School Java Programming Overview

Course Description	Topics at a Glance
By design, this course enables middle school students, with no previous experience, to jump right into the Java programming language. Students will become familiar with objects, classes, methods, parameters, data types, fields, loops, constructors and more. Students will learn about these topics, as well as others, through modifying existing programs and creating programs from scratch.	• Objects, Classes, & Methods • Fields, Constructors, Parameters • Abstraction & Modularization • Conditional Statements & Loops • Collection Classes • Parameters & Variables • Java Class Libraries
Assessments • Written Exams • Computer-Based Exams • Project-Based Assessments	Achieving these goals may be reached by a variety of projects and/or programs. Use of specific software programs or technology equipment were deliberately not mentioned due to the variation in resources among different schools. **Standards:** The National Education Technology Standards (NETS) for Students were developed in 1998 and updated in 2007 by ISTE, the International Society for Technology in Education. 1. Creativity and innovation 2. Communication and Collaboration 3. Research and Information Fluency 4. Critical Thinking, Problem Solving, and Decision Making 5. Digital Citizenship 6. Technology Operations and Concepts 7. Careers

Content Area: Technology – Middle Level Java Programming

Standard 1. Creativity and Innovation:
Students demonstrate creative thinking, construct knowledge, and develop original programs.

Prepared Graduates:
Apply existing knowledge to generate new ideas, products, or processes. Create original works as a means of personal or group expression. Use models and simulations to explore complex systems and issues. Identify trends and forecast possibilities.

GRADE LEVEL EXPECTATION
Concepts and skills students master:
Apply existing knowledge to generate new ideas, products, or processes.
Create original works as a means of personal or group expression.
Use models and simulations to explore complex systems and issues.
Identify trends and forecast possibilities.

Evidence Outcomes	21st Century Skills and Readiness Competencies
Students can: a. Apply prior learning to develop innovative solutions to problems. b. Produce a checklist of design constraints and apply them to a given project. c. Test, modify, and retest, solutions to produce an improved final program. d. Provide documentation of the process leading to a solution.	**Inquiry Questions:** 1. How is creativity different from problem solving? 2. Why are there always tradeoffs in design problems? 3. What are some of the ethical responsibilities that a designer should consider? **Relevance and Application:** 1. Program design involves trade-offs among competing constraints and requirements. **Nature of Discipline:** 1. Work in teams to solve a given problem.

Content Area: Technology – Middle Level Java Programming

Standard: 2. Communication and Collaboration:
Students work cooperatively to brainstorm possible solutions to multi-faceted problems programming problems. Students work together to choose and apply potential solutions. Students will present solutions to peers and teachers.

Prepared Graduates:
Communicate information and ideas effectively to multiple audiences. Contribute to team projects to produce original works or solve problems.

GRADE LEVEL EXPECTATION
Concepts and skills students master:
Communicate information and ideas effectively to multiple audiences.
Contribute to team projects to produce original works or solve problems.

Evidence Outcomes	21st Century Skills and Readiness Competencies
Students can: a. Communicate effectively with others during brainstorming sessions. b. Successfully complete group projects and demonstrate cooperation, teamwork, and division of labor to complete the assigned task. c. Successfully communicate evidence of successful programming.	**Inquiry Questions:** 1. What skills do we need to work effectively with others? 2. How might people collaborate in the future? 3. Why is working with others so important?
	Relevance and Application: 1. Communication and cooperation are necessary in everyday life. These skills will remain necessary in our future careers as well.
	Nature of Discipline: 1. Having skills to complete tasks and communicate solutions as an individual and as a team member is essential for the 21st Century Graduate.

Content Area: Technology – Middle Level Java Programming

Standard: 3. Research and Information Fluency:
Students use digital tools to gather, evaluate, and use information.

Prepared Graduates:
Plan strategies to guide inquiry. Locate, organize, analyze, evaluate, synthesize, and ethically use information from a variety of sources. Process Data and report results.

GRADE LEVEL EXPECTATION
Concepts and skills students master:
Plan strategies to guide inquiry.
Locate, organize, analyze, evaluate, synthesize, and ethically use information from a variety of sources.
Process Data and report results.

Evidence Outcomes	21st Century Skills and Readiness Competencies
Students can: a. Plan design strategies and processes by creating a project management document listing steps in the design process, a timeline for completion, and division of workload. b. Locate, analyze, and evaluate Java exemplars to devise strategies and formulate programming questions and problems. c. Evaluate and select appropriate components to be used to meet the program specifications. *d.* Test solutions and compile a report to communicate the results of the collected data.	**Inquiry Questions:** 1. What role does research play in the programming process? 2. How can we assess/measure the quality of a program? **Relevance and Application:** 1. Technology drives invention and innovation as a dynamic process. **Nature of Discipline:** 1. Project-based design and implementation

Content Area: Technology – Middle Level Java Programming

Standard: 4. Critical Thinking, Problem Solving, and Decision Making:
Students use critical thinking skills to design programs, manage projects, solve problems, and make informed decisions using appropriate tools and resources.

Prepared Graduates:
Identify and define authentic problems and significant questions for investigation. Plan and manage activities to develop a solution or complete a project. Collect and analyze data to identify solutions and/or make informed decisions. Use multiple processes and diverse perspectives to explore alternative solutions.

GRADE LEVEL EXPECTATION
Concepts and skills students master:
Identify and define authentic problems and significant questions for investigation.
Plan and manage activities to develop a solution or complete a project.
Collect and analyze data to identify solutions and/or make informed decisions.
Use multiple processes and diverse perspectives to explore alternative solutions.

Evidence Outcomes	21st Century Skills and Readiness Competencies
Students can: a. Identify and define project goals and intentions by describing the problem or challenge. b. Define the strategies they will use to meet project goals while addressing constraints. c. Explore and refine programming solutions by starting with pseudo-code. d. Assess strengths and weaknesses of alternate design solutions. e. Demonstrate evidence of creativity in the programming process and final product.	**Inquiry Questions:** 1. How and why do we use the scientific process? 2. What are the steps or stages of a typical scientific process? **Relevance and Application:** 1. Technological design is a systematic process used to initiate and refine ideas, solve problems, and maintain products and systems. **Nature of Discipline:** 1. Design and engineering.

Content Area: Technology – Middle Level Java Programming

Standard: 5. Digital Citizenship:
Students understand human cultural, and societal issues related to computer technology and practice legal, ethical behavior.

Prepared Graduates:
Advocate and practice safe, legal, and responsible use of information and technology. Exhibit a positive attitude toward using technology that supports collaboration, learning, and productivity. Demonstrate personal responsibility for lifelong learning.

GRADE LEVEL EXPECTATION
Concepts and skills students master:
Advocate and practice safe, legal, and responsible use of information and technology.
Exhibit a positive attitude toward using technology that supports collaboration, learning, and productivity.
Demonstrate personal responsibility for lifelong learning.

Evidence Outcomes	21st Century Skills and Readiness Competencies
Students can: a. Demonstrate knowledge of copyright, patent, and fair practices. b. Illustrate the impact programming can have on quality of life. c. Describe ethical and human impact issues in programming through classroom discussion. d. Exhibit personal initiative, leadership, and responsibility.	**Inquiry Questions:** 1. Is the impact of computers/software on society positive in all aspects? 2. What is the relationship between technology and quality of life? 3. What are the implications of constantly evolving computer related technologies?
	Relevance and Application: 1. Advances in technology can impact us in both positive and negative ways. 2. Technology can be used to improve our quality of life. 3. There are inherent risks that come with the use of certain technologies.
	Nature of Discipline: 1. How technology impacts our lives will become more and more prevalent for a 21st century graduate.

Content Area: Technology – Middle Level Java Programming

Standard: 6. Technology Operations and Concepts:
Students demonstrate a sound understanding of technology concepts, systems, and operations.

Prepared Graduates:
Understand and use technological systems. Select and use applications effectively and productively.

GRADE LEVEL EXPECTATION
Concepts and skills students master:
Understand and use technological systems through the creation of Java programs.
Select and use hardware and software effectively and productively.

Evidence Outcomes	21st Century Skills and Readiness Competencies
Students can: a. Correctly identify, categorize, and use project-relevant resources. b. Show evidence of operational skills when using computers and other related technologies. c. Demonstrate safe use of all resources.	**Inquiry Questions:** 1. How do I use our technology responsibly and safely? **Relevance and Application:** 1. The acquisition of knowledge is needed to use and operate the various technologies used in Java Programming environments. **Nature of Discipline:** 1. Using tools and equipment to design and build projects with real world application potential.

Glossary of Terms

Word	Definition
abstraction	A simplified representation of something that is potentially quite complex. It is often not necessary to know the exact details of how something works, is represented or is implemented, because we can still make use of it in its simplified form. Object-oriented design often involves finding the right level of abstraction at which to work when modeling real-life objects. If the level is too high, then not enough detail will be captured. If the level is too low, then a program could be more complex and difficult to create and understand than it needs to be.
accessor method	A method specifically designed to provide access to a private attribute of a class. By convention, we name accessors with a get prefix followed by the name of the attribute being accessed. For instance, the accessor for an attribute named speed would be getSpeed. By making an attribute private, we prevent objects of other classes from altering its value other than through a mutator method. Accessors are used both to grant safe access to the value of a private attribute and to protect attributes from inspection by objects of other classes. The latter goal is achieved by choosing an appropriate visibility for the accessor.
applet	Applets are Java programs based around the Applet or JApplet classes. They are most closely associated with the ability to provide active content within Web pages. They have several features which distinguish them from ordinary Java graphical applications, such as their lack of a user-defined main method, and the security restrictions that limit their abilities to perform some normal tasks.
application	Often used simply as a synonym for program. However, in Java, the term is particularly used of programs with a Graphical User Interface (GUI) that are not applets.
application programming interface (API)	A set of definitions that you can make use of in writing programs. In the context of Java, these are the packages, classes, and interfaces that can be used to build complex applications without having to create everything from scratch.
argument	Information passed to a method. Arguments are also sometimes called parameters. A method expecting to receive arguments must contain a formal argument declaration for each as part of its method header. When a method is called, the actual argument values are copied into the corresponding formal arguments.
arithmetic expression	An expression involving numerical values of integer or floating point types. For instance, operators such as +, -, *, / and %take arithmetic expressions as their operands and produce arithmetic values as their results.
arithmetic operator	Operators, such as +, -, *, / and %, that produce a numerical result, as part of an arithmetic expression.
array	A fixed-size object that can hold zero or more items of the array's declared type. The initializer takes the place of separate creation and initialization steps.
assignment operator	The operator (=) used to store the value of an expression into a variable
assignment statement	A statement using the assignment operator.
behavior	The methods of a class implement its behavior. A particular object's behavior is a combination of the method definitions of its class and the current state of the object.
block	Statements and declarations enclosed between a matching pair of curly brackets

	({ and }). For instance, a class body is a block, as is a method body. A block encloses a nested scope level.
boolean	One of Java's primitive types. The boolean type has only two values: true and false.
boolean expression	An expression whose result is of type boolean, i.e. gives a value of either true or false. Operators such as && and || take boolean operands and produce a boolean result. The relational operators take operands different types and produce boolean results.
cast	Where Java does not permit the use of a source value of one type, it is necessary to use a cast to force the compiler to accept the use for the target type. Care should be taken with casting values of primitive types, because this often involves loss of information. Casts on object references are checked at runtime for legality. A ClassCastException exception will be thrown for illegal ones.
Central Processing Unit	The Central Processing Unit (CPU) is the heart of a computer as it is the part that contains the computer's ability to obey instructions. Each type of CPU has its own instruction set.
class	A programming language concept that allows data and methods to be grouped together. The class concept is fundamental to the notion of an object-oriented programming language. The methods of a class define the set of permitted operations on the class's data (its attributes). This close tie between data and operations means that an instance of a class - an object - is responsible for responding to messages received via its defining class's methods.
class body	The body of a class definition. The body groups the definitions of a class's members -fields, methods and nested classes.
class header	The header of a class definition. The header gives a name to the class and defines its access. It also describes whether the class extends a super class or implements any interfaces.
class method	A synonym for static method.
class scope	Private variables defined outside the methods within a class have class scope. They are accessible from all methods within the class, regardless of the order in which they are defined. Private methods also have class scope. Variables and methods may have a wider scope if they do not use the private access modifier.
class variable	A synonym for static variable.
command-line argument	Arguments passed to a program when it is run. A Java program receives these in the single formal argument to its main method
comment	A piece of text intended for the human reader of a program. Compilers ignore their contents.
compilation	The process of translating a programming language. This often involves translating a high level programming language into allow level programming language, or the binary form of a particular instruction set. The translation is performed by a program called a compiler. A Java compiler translates programs into byte codes.
compiler	A program which performs a process of compilation on a program written in a high level programming language.
condition	A boolean expression controlling a conditional statement or loop.
data type	There are eight primitive data types in Java; five of these represent numerical types of varying range and precision - double,float, int, long and short. The

	remaining three are used to representing single-bit values (boolean), single byte values (byte) and two-byte characters from the ISO Unicode character set (char).
declaration & initialization	A statement in which a variable is declared and immediately given its initial value.
do loop	One of Java's three control structures used for looping. The other two are the while loop and for loop. A do loop consists of a loop body and a boolean expression. The condition is tested after the loop body has been completed for the first time and re-tested each time the end of the body is completed. The loop terminates when the condition gives the value false. The statements in the loop body will always be executed at least once.
field	Variables defined inside a class or interface, outside of the methods. Fields are members of a class.
for loop	One of Java's three control structures used for looping. The other two are the while loop and do loop. A for loop consists of a loop header and a loop body. The header consists of three expressions separated by two semicolons and one or more of these may be omitted. The first expression is only evaluated once, at the point the loop is entered. The middle expression is a boolean expression representing the loop's termination test. An empty expression represents the value true. The third expression is evaluated after each completion of the loop's body. The loop terminates when the termination test gives the value false. The statements in the loop body might be executed zero or more times.
global variable	A phenomenon that is more usually regarded as being a problem in structured programming languages than in object-oriented languages. In a structured programming language, such as Pascal or C, a global variable is one defined outside the procedures and functions of a program. It is difficult to keep track of the usage of such a variable as it is readable and writable by the whole program or module in which it is defined. This makes such variables a common source of logical errors. In fact, instance variables pose a similar problem within class definitions, since Java's scope rules make them accessible to all methods defined within a class. This is one of the reasons why we prefer to channel access to instance variables through accessor and mutator methods even within a class.
graphical user interface	A Graphical User Interface (GUI) is part of a program that allows user interaction via graphical components, such as menus, buttons, text areas, etc. Interaction often involves use of a mouse.
if-else statement	A control structure used to choose between performing one of two alternative actions.
if statement	A control structure used to choose between performing or not performing further actions.
import statement	A statement that makes the names of one or more classes or interfaces available in a different package from the one in which they are defined. Import statements follow any package declaration {package!declaration}, and precede any class or interface definitions.
infinite loop	A loop whose termination test never evaluates to false. Sometimes this is a deliberate act on the part of the program
inheritance	A feature of object-oriented programming languages in which a sub type inherits methods and variables from its super type. Inheritance is most

	commonly used as a synonym for class inheritance {class!inheritance}, but interface inheritance is also a feature of some languages, including Java.
instance	A synonym for object. Objects of a class are instantiated when a class constructor is invoked via the new operator.
instance variable	A non-static field of a class. Each individual object of a class has its own copy of such a field. This is in contrast to a class variable which is shared by all instances of the class. Instance variables are used to model the attributes of a class.
instantiation	The creation of an instance of a class - that is, an object.
integer	A positive or negative whole number. The primitive types byte, short, int and long are used to hold integer values within narrower or wider ranges.
iteration	Repetition of a set of statements, usually using a looping control structure, such as a while loop, for loop or do loop.
Java	A portable high level programming language released by Sun Microsystems (now Oracle).
Java Virtual Machine (JVM)	An idealized machine whose instruction set consists of byte codes. A Java program is compiled to an equivalent byte code form and executed on an interpreter which implements the JVM.
key value	The object used to generate an associated hash code for lookup in an associative data structure.
local variable	A variable defined inside a method body.
logical operators	Operators, such as &&, ||, &, | and ^ that take two boolean operands and produce a boolean result. Used as part of a boolean expression, often in the condition of a control structure.
loop variable	A variable used to control the operation of a loop, such as a for loop. Typically, a loop variable will be given an initial value and it is then incremented after each iteration until it reaches or passes a terminating value.
main method	The starting point for program execution public static void main(String[] args)
method	The part of a class definition that implements some of the behavior of objects of the class. The body of the method contains declarations of local variables and statements to implement the behavior. A method receives input via its arguments, if any, and may return a result if it has not been declared as void.
method body	The body of a method: everything inside the outermost block of a method.
method header	The header of a method, consisting of the method name, its result type, formal arguments and any exceptions thrown. Also known as a method signature.
method overloading	Two or more methods with the same name defined within a class are said to be overloaded. This applies to both constructors and other methods. Overloading applies through a class hierarchy, so a sub class might overload a method defined in one of its super classes. It is important to distinguish between an overloaded method and an overridden method. Overloaded methods must be distinguishable in some way from each other; either by having different numbers of arguments, or by the types of those arguments being different. Overridden methods have identical formal arguments.
method result	The value returned from a method via a return statement. The type of the expression in the return statement must match the return type declared in the method header.

method signature	A synonym for method header.
mutator method	A method specifically designed to allow controlled modification of a private attribute of a class. By convention, we name mutators with a set prefix followed by the name of the attribute being modified. For instance, the mutator for an attribute named speed would be setSpeed. By making an attribute private, we prevent objects of other classes from altering its value other than through its mutator. The mutator is able to check the value being used to modify the attribute and reject the modification if necessary. In addition, modification of one attribute might require others to be modified in order to keep the object in a consistent state. A mutator method can undertake this role. Mutators are used both to grant safe access to the value of a private attribute and to protect attributes from modification by objects of other classes. The latter goal is achieved by choosing an appropriate visibility for the mutator.
newline	The \n character.
new operator	The operator used to create instances {instance} of a class.
null reference	A value used to mean, `no object'. Used when an object reference variable is not referring to an object.
object	An instance of a particular class. In general, any number of objects may be constructed from a class definition (see singleton, however). The class to which an object belongs defines the general characteristics of all instances of that class. Within those characteristics, an object will behave according to the current state of its attributes and environment.
object-oriented language	Programming languages such as C++ and Java that allow the solution to a problem to be expressed in terms of objects which belong to classes.
operand	An operand is an argument of an operator. Expressions involve combinations of operators and operands. The value of an expression is determined by applying the operation defined by each operator to the value of its operands.
operator	A symbol, such as -, == or ?: taking one, two or three operands and yielding a result. Operators are used in both arithmetic expressions and boolean expressions.
out-of-bounds value	A redundant value used to indicate that a different action from the norm is required at some point. The read method of InputStream returns -1 to indicate that the end of a stream has been reached, for instance, instead of the normal positive byte-range value.
package	A named grouping of classes and interfaces that provides a package namespace. Classes, interfaces and class members without an explicit public, protected or private access modifier {access!modifier} have package visibility. Public classes and interfaces may be imported into other packages via an import statement.
parameter	See argument.
parsing	Usually applied to the action of a compiler in analyzing a program source file for syntax errors. It is also used more widely to mean the analysis of the structure of input.
relational operators	Operators, such as <, >, <=, >=, == and !=, that produce a boolean result, as part of a boolean expression.
primitive type	boolean, byte, char, double, float, int, long and short
return statement	A statement used to terminate the execution of a method.

return type	The declared type of a method, appearing immediately before the method name.
return value	The value of the expression used in a return statement.
scope	A language's scope rules determine how widely variables, methods and classes are visible within a class or program. Local variables have a scope limited to the block in which they are defined, for instance. Private methods and variables have class scope, limiting their accessibility to their defining class. Java provides private, package, protected and public visibility.
single-line comment	single line comment
software	Programs written to run on a computer.
software engineering	The system of applying of an engineering discipline to the design, implementation and maintenance of software systems.
state	Objects are said to possess state. The current state of an object is represented by the combined values of its attributes. Protecting the state of an object from inappropriate inspection or modification is an important aspect of class design and we recommend the use of accessor methods and mutator methods to facilitate attribute protection and integrity. The design of a class is often an attempt to model the states of objects in the real-world. Unless there is a good match between the data types available in the language and the states to be modeled, class design may be complex. An important principle in class design is to ensure that an object is never put into an inconsistent state by responding to a message.
statement	The basic building block of a Java method. There are many different types of statement in Java, for instance, the assignment statement, if statement, return statement and while loop.
string	An instance of the String class. Strings consist of zero or more Unicode characters, and they are immutable, once created. A literal string is written between a pair of string delimiters (").
syntax error	An error detected by the compiler during its parsing of a program. Syntax errors usually result from mis-ordering symbols within expressions and statements. Missing curly brackets and semicolons are common examples of syntax errors.
variable declaration	The association of a variable with a particular type. It is important to make a distinction between the declaration of variables of primitive types and those of class types. A variable of primitive type acts as a container for a single value of its declared type. Declaration of a variable of a class type does not automatically cause an object of that type to be constructed and, by default, the variable will contain the value null. A variable of a class type acts as a holder for a reference to an object that is compatible with the variable's class type. Java's rules of polymorphism allow a variable of a class type to hold a reference to any object of its declared type or any of its sub types. A variable with a declared type of Object, therefore, may hold a reference to an object of any class
while loop	One of Java's three control structures used for looping. The other two are the do loop and for loop. A while loop consists of a boolean expression and a loop body. The condition is tested before the loop body is entered for the first time and re-tested each time the end of the body is completed. The loop terminates when the condition gives the value false. The statements in the loop body might be executed zero or more times.
white space	Characters used to create visual spacing within a program. White spaces include

	space, tab, carriage return and line feed characters.