'Programming-Fundamentals-Assignment-2'

1.	Statically type languages
	In statically typed programming languages, type checking occurs at compile time. At compile time, source code in a specific programming language is converted to a machine-readable format. This means that before source code is compiled, the type associated with each and every single variable must be known.
	Dynamically type languages
	In **dynamically typed languages,** type checking takes place at runtime or execution time. This means that variables are checked against types only when the program is executing. Some examples of programming languages that belong to this category are Python, JavaScript, Lisp, PHP, Ruby, Perl, Lua, and Tcl.
	Strongly typed languages
	Strongly typed languages are written based on the highly considering the data types
	Loosely Typed languages
	Loosely typed languages do not consider the about the data type to assigning variables and objects
	Java - Java is combinationof statistically and dynamically typed langagage. Its highly consider about the data types so it is a Strongly typed language
2.	**Case Sensitive** Case sensitive means strongly consider about the case of the wording . It describe the ability to distinguish between upper and lower case versions of letters of language words.
	Case Insensitive Case insensitive means languages that cannot distinguish the difference of the upper and lower cases of the words
	Case Sensitive -Insensitive
3.	**Concept of identity Conversion**
	The concept of identity conversion, also known as identity casting, refers to a type conversion in computer programming where a value is cast to the same data type it already belongs to. In other words, identity conversion does not change the underlying representation or value of the data; it simply allows the programmer to assert that the value is of the same type as the target variable explicitly.

	operators for the sake of clarity. **String X=”Hello”;** **String Y= X;**
4.	Primitive widening Conversion
	Prmitive widening conversion means data type with small storing capacity is converting in to data type with large strong capacity. ● byte to short, int, long, float, or double ● short to int, long, float, or double ● char to int, long, float, or double ● int to long, float, or double ● long to float or double ● float to double
5.	**Compile type constants** - compiler knows inside the value when it compiling **Run type constants** - in here compiler don’t know the value when they compile. JVM knows when it runs
6. i	**Implicit narrowing primitive conversion** implicit narrowing conversion, is a type of data conversion in computer programming, particularly in the context of primitive data types. It occurs when a value of a larger data type is assigned to a variable of a smaller data type without requiring any explicit type casting or conversion operator. The value is automatically truncated or reduced to fit within the target variable's range.
	Explicit narrowing primitive conversion Explicit narrowing primitive conversion, also known as explicit type casting, is a type of data conversion in computer programming that requires the programmer to explicitly specify the conversion from a larger data type to a smaller data type. This is done using a type cast, indicating that the programmer is aware of the potential loss of data precision and explicitly intends to perform the conversion. Explicit narrowing conversion is necessary when assigning a value of a larger data type to a variable of a smaller data type, where the value may not fit within the range of the target variable without loss of data
7.	**A widening primitive conversion from int to float, or from long to float, or from long to double,** may result in loss of precision - that is, the result may lose some of the least significant bits of the value. In this case, the resulting floating-point value will be a correctly rounded version of the integer value, using IEEE 754 round-to-nearest mode
8.	**Why are int and double set as the default data types for integer literals and floating point literals respectively in Java?**
	Chapter 3: Lexical Structure Section 3.10.1: Integer Literals Section 3.10.2: Floating-Point Literals

	Chapter 3 of the JLS deals with the lexical structure of the Java programming language, which includes the rules for writing literals such as integers and floating-point numbers. In Section 3.10.1, it explains the different formats for writing integer literals in Java, including decimal, octal, hexadecimal, and binary. It also specifies that the default data type for an integer literal without any suffix is int. In Section 3.10.2, it covers the formats for writing floating-point literals, such as decimal and hexadecimal floating-point literals. It also specifies that the default data type for a floating-point literal without any suffix is double.
9.	A narrowing conversion of a signed integer to an integral type T simply discards all but the n lowest order bits, where n is the number of bits used to represent type T. In addition to a possible loss of information about the magnitude of the numeric value, this may cause the sign of the resulting value to differ from the sign of the input value. A narrowing conversion of a char to an integral type T likewise simply discards all but the n lowest order bits, where n is the number of bits used to represent type T. In addition to a possible loss of information about the magnitude of the numeric value, this may cause the resulting value to be a negative number, even though chars represent 16-bit unsigned integer values.
10.	A narrowing primitive conversion may lose information about the overall magnitude of a numeric value and may also lose precision and range. Minimizing potential data loss: Implicit narrowing conversions can lead to data loss when converting larger data types to smaller ones. By limiting the implicit conversions to these four data types, Java helps ensure that developers are aware of the possible data loss and are more likely to use explicit type casting when necessary.