1. Programming Paradigms:

Declarative

Functional

Imperative

Logic

Object-oriented

Symbolic

1. What are some of the different types of languages

Statically Typed:compile time

Dynamically Typed:runtime

Manifestly Typed

Type Infferred

Strongly Typed

Weakly Typed

Java is a manifestly strongly typed language

1. What is the difference between compiled and interpreted

Compiler convert source code to binary machine code first. It’s platform dependent. It cannot be transferred from one platform to another.

The interpreter will read this code and perform the instructions. These script can be run on any platform where the interpreter exists.

1. What type of language is Java

Manifestly, strongly typed and both compiled and interpreted language.

1. What is the JVM

An abstract machine that interprets and executes Java Byte Code.

1. Delivery Process

Develop source code-> compiler converts source code to java byte code->Java byte code is distributed to different client -> JVM interprets the JAVA Byte code-> java program executes.

1. What commands would you need to take Java source code and execute it?

Javac

Java

1. Benefit of IDE

IDE: Integrated Development Environment

Automation tool which performs many normal tasks automatically ( Compilation, Project Set up)

Provides code suggestions

Error warnings

Automates refactoring

Automates configuration

1. What are the benefits to being able to code without an IDE?

There are some cases that we may not be able to use an IDE, in that case, we still need to know how to write Java code in the note book and how to compile it, execute it.

1. What is the Best Practices?
   1. PascalCase classes
   2. camelCase variables and methods
   3. Good naming of variables and methods, should be self-descriptive
   4. Comment only where applicable
   5. Indent your code correctly
2. The benefit of best practices?

Save you and others time and frustration.

1. What is the syntax of the definition of a method?

Access Modifier Return data type Method name braket parameters

1. What is the definition syntax of a class?

Access Modifier class

1. What is a Packages?

A namespace is an area where each item must have a unique name

In Java a package is a namespace for Java files. They also appear on disk as folders.

This prevents class/interface from having the same name in the same namespace. If they are located in a separate namespace, then they can have a duplicate name.

1. What is the syntax of a package declaration?

Must be in the first line of the file.

Any import statement must go directly after it.

1. What is the Access Modifier:

To declare the visibility of a method/variable to other classes.

The **default**, **package-private**, or **package**access level

Any member of class A without a modifier can be seen from all classes in the same package as A

1. What is Final

Prevents extension.

Class: may not be extended.

Methods: May not be overridden

0Variables: May not be assigned a new value

1. What is Static?

A static method or variable belongs to the class, rather than an instance of a class.

1. What flow control is permitted?

If

Switch

Temary operator **result = (a > b) ? a : b;**

While loop

Do while loop

For loop

For each

Constructor

1. What is a Constructor

Block of code that runs when an object is created

* Used to initialize the object’s state and prepare it for use
* Runs before any methods are invoked or any fields are accessed
* The process is invoked with the keyword **new**

1. What is difference?

No return type

Must have same name as class

1. What is the first statement in the constructor

Every constructor must begin with a call to **this()** or **super()**

* + These calls can have arguments
  + If neither is called, the compiler inserts a call to super()

**super()** calls a constructor in the parent class

**this()** calls another constructor in the same class

* + Often done to avoid duplicating code
  + Not to be confused with **this** (Java keyword)

1. Constructor are not inherited
2. What is the constructor chaining?

the sequence of constructors invoked through calls to this() and super()

* + Chain must be able to complete in order to compile

1. What are primitive?

Primitive variables hold literal values

Manipulated using Java operator(not methods)

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Size** | **Default Value** | **Range** |
| **byte** | 8 bit | 0 | -128 to 127 |
| **short** | 16 bit | 0 | -32,768 to 32,767 |
| **int** | 32 bit | 0 | -2,147,483,648 to 2,147,483,647 |
| **long** | 64 bit | 0L | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| **float** | 32 bit | 0.0F | (+-) 1.40129846432481707e-45 to  (+-) 3.40282346638528860e+38 |
| **double** | 64 bit | 0.0 | (+-) 4.94065645841246544e-324 to  (+-) 1.79769313486231570e+308 |
| **boolean** | undefined | false | false to true |
| **char** | 16 bit | '\u0000' | '\u0000' to '\uffff' |

Numeric Literal: can be formatted with \_ to improve readability

1. What is reference type

Class:

Contain both data (attributes) and behaviours (methods)

Provide blueprints which can be instantiated later

Allows you to create custom data types

Many classes are built into the Java library

Objects :

are instances of classes

Created using the *new* keyword

All classes in java extend the Object class

1. What is String:

Reference type

Immutable

String Literals are stored in the String Pool

But you can create a new String object with new.

**How many Strings are getting Created in the String Pool?**

Sometimes in java interview, you will be asked a question around String pool. For example, how many strings are getting created in the below statement;

String str = new String("Cat");

In the above statement, either 1 or 2 string will be created. If there is already a string literal “Cat” in the pool, then only one string “str” will be created in the pool. If there is no string literal “Cat” in the pool, then it will be first created in the pool and then in the heap space, so a total of 2 string objects will be created.

1. String alternative

StringBuffer

StringBuilder

1. What is an Arrays

A way of containing multiple variable of the same type

Reference

Immutable

Indexed form 0

Syntax uses square bracket

1. What is Enum

A way to represent a set of predefined values

* + Essentially constants, represented in a readable and meaningful way
  + Act as a flag which can be read and utilised in conditional logic
  + More efficient than using Strings

1. What is Wrappers
   1. Classes that wrap a primitive value and provide additional functionality
   2. Wrappers for numeric types extend the Number class
2. Big decimal

Fixes the precision issues with float and double

1. What are some of the key utility classes and their benefits?

Big decimal

Math Class: contains several fields and methods for different mathematical operations

https://www.liaoxuefeng.com/wiki/1252599548343744/1279768011997217

1. What is an abstract class

Cannot be initiated

* Marked with **abstract** modifier
* Can contain instance/static member variables
* Can contain instance/static method implementations
* Can contain *abstract methods*

1. What is an abstract method

Has no method body

Can only belong to an abstract class or interface

A concrete class extends a abstract class must implement all the abstract methods.

1. What is an interfaces

An interface describes how the outside world can interact with a class.

* + All methods are implicitly **public** and **abstract**
  + All fields are implicitly **public** **static** and **final**

1. When a class implements an interface:
   * It guarantees to provide the behavior listed by the interface
   * It gains the data type that the interface defines (IS-A)
2. What is default

The default keyword allows default implementation of methods in interfaces

Use default keyword before return type in method signature to allow implementation body

Implementing classes do not have to override – default implementation will be used

Implementing classes are able to override method if behavior needs to be changed

1. Static in interfaces

Allows static methods in interfaces

Belong to the interface; can only be seen and called through the name of the interface

Functionally identical to static methods in classes

1. Difference between abstract class and interface?

Abstract class: can define base behavior with concrete methods.

Part of the class hierarchy

Can only have one parent class

Interfaces: List guaranteed behavior

Multiple interfaces can be implemented at once

Describe what an object can do

1. What is value type

Pieces of memory holding a raw value primitive variable

1. What is Reference type

Aliases to location in memory where the actual value is

* + object variables are reference types

1. Pass by value vs pass by reference Litmus Test

Pass by Value:

* + The variable’s value is evaluated
  + A **copy** is created
  + The **copy** is assigned to the new variable

Pass by Reference:

* + New variable receives a reference to the original variable
  + Any change to the new variable modifies the original variable

1. What is in stack?

Method calls and local variables.

Primitives

Objects reference

1. Heap

All objects and their data

Only one heap in the JVM

1. What does the garbage collector do?

Part of the JVM that reclaims and maintains memory on the Heap

* + Runs in its own Thread
  + Cannot be pre-emptied
  + Stops the application completely while it runs
  + Main reason why Java cannot be used in real time situations

1. What is encapsulation

Groupping

High Cohesion

Surround and cover to restricted access and hide implementation details

Low coupling

The act of grouping together data and behavior into logical components.

1. What is abstraction?

Extracting relevant and essential information and behavior.

1. Inheritance

The act of one class receiving the behaviours and attributes of another class or interface.

IS-A relationship

1. What is Polymorphism

Having multiple forms or versions of the same type of thing and the same types of behaviors.

1. What is Binding

Joining attributes and method calls to actual objects and behaviors in the memory

How the program decides on the next action or piece of functionality which needs executing.

1. What is Early binding and what is late binding?

Early Binding——Overloading

Attributes and method calls are bound when compiling source code

Late Binding——Override

Attributes and method calls are bound when the program is running

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or [objects](https://searchapparchitecture.techtarget.com/definition/object), rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

OOP is a way of building software

It is modular, with the class as the primary building block

1. What is autoboxing/unboxing?

Convert between primitive types and object types.

1. Why we need autoboxing?

Primitives could not be added to collections

Primitives could not be passed into methods that take an Object

1. What is unboxing?

Implicitly extracting a primitive type from a wrapper type

1. What is widening?

Widening is taking place when a small primitive type value is automatically accommodated in a bigger/wider primitive data type

Widening is the way the compiler resolves overloading ambiguity.

The compiler picks the method with the smallest parameter   
that can hold the passed-in data type.

1. Varargs

Notice how format() can take in any number of arguments.

In Java, variable argument methods use the following syntax:

* Gets passed in as an array
* Can mix with other parameters, but only one var-arg is allowed
* Must always appear last in the argument list

1. Casting

Casting allows us to treat objects as different, but compatible types:

Two types:

* + Upcasting (implicit casting) subclass->parentclass
  + Downcasting (explicit casting) parentclass->subclass

*must*reference an OptionTrade object.

* + Otherwise, a ClassCastException will be thrown

The compiler alone cannot verify this, but there is a way to perform this check at runtime

1. What is a collection?

A collection is a way of grouping objects of commonality together

1. What are the main interfaces in the Collections Framework?

List Queue Set

1. What are the benefits of using an ArrayList versus a LinkedList?

Fast Iteration

Random access

1. What are the benefits of using a HashMap?
   1. Defines an object that maps **keys** to **values**
   2. Each key maps to only one value, duplicate keys are not allowed, duplicate values are allowed
   3. Part of the *Collections Framework*, but does not extend Collection
2. What is the purpose of a Queue?

FIFO/LIFO collections

1. What are the ways we can compare two objects to each other?

Comparable compareTo

Comparator compareTo and equals

1. What is the hashCode() method used for? What about equals()?
   * Used to determine whether an object is equal to another

Returns an int hash code value for any given object

* + Also defined in the Object class and can be overridden
  + Typically based on the same things being compared in equals

1. What class contains sorting and ordering utility methods?

TreeMap

TreeSet

PriorityQueue

**Collections** is a class that provides functionality for:

* + Sorting and ordering
  + Searching
  + Converting
  + Min and Max entries
  + Singleton collections
  + Empty collections
  + Extracting elements within a collection
  + Many algorithms needed to deal with Collection objects

Class Polymorphism: subclass

Method Polymorphism: method overloading

What is the extension of the java byte code: .class

CompareTo

Liked transfer queue

LinkedList order

TransferQueue FIFO Trransfer

1. What is a Generic?

Generics allow you to specify the type of object that another object should deal with in a dynamic way.

1. What is a Generic class?

Add compile-time type safely to class.

A class

1. What is a type parameter?

TypeParameter is a frequently used generic label used in templates to reference an unknown data type

1. Where might Generics be useful?

Typically used for Collections or storage mechanisms

Possible scenarios include:

* + Command Pattern, meaning each command could work with a specific type of object
  + Storage for Databases, meaning each Storage class could store a specific type of object
  + New Collection functionality, each Collection holds the same type of object

1. What syntax do you need to create Generics?

<T t>

1. How are bounds used with generics?

Bounded Type Parameters

There may be times when you'll want to restrict the kinds of types that are allowed to be passed to a type parameter. For example, a method that operates on numbers might only want to accept instances of Number or its subclasses. This is what bounded type parameters are for.

To declare a bounded type parameter, list the type parameter's name, followed by the extends keyword, followed by its upper bound.

1. What is a wildcard?

Wildcards give us more flexibility when creating references to generic types:

We use wildcard ? instead of the T when we want to creating references to generic types.

Clean Code

Do you need a comment?

Meaningful identifiers can replace unclear code and take away the need to comment**.**

What is good commenting?

|  |  |
| --- | --- |
| **Usage** | **For Example…** |
| Legal Purposes | Copyright or authorship comments. |
| Warning of Consequences | Ex. Not thread safe, takes long to run, etc. |
| TODO Comments | Reminders to do something that cannot be handled at the moment. |
| JavaDocs | Comments for public API. |
| Amplification | Drawing attention to importance of something. |
| Informative | Explanation of a regular expression, or format string (hard to make readable on its own). |
| Explanation of Intent | The intent behind a feature or an implementation. |
| Clarifying Foreign Code | Clarification of code you cannot alter to make more readable. |

Identifier

|  |  |
| --- | --- |
| **The Good** | **The Bad** |
| Intention-Revealing | Noise Words |
| Pronounceable | Encoded Names |
| Searchable | “Mental Mapping” |
| Nouns for Class Names | Misleading |
| Verbs for Method Names | Prefixes |
| Proper Domain Terminology | Puns |
| Added Meaningful Context | Trying to be clever |

Class Organization

* Variables (standard Java convention)
  1. First, public static constants
  2. Then, private static variables
  3. Next, private instance variables
* There is seldom good reason to have a public variable
* Public functions should follow the list of variables

Data Hiding

* Keep variables and utility functions private
* Testability is important
  1. Protected/package access might be needed for testing
* Always look for a way to maintain privacy
  1. Loosening encapsulation is a last resort!

SOLID

从功能上来看，接口隔离和单一职责两个原则具有一定的相似性。其实如果我们仔细想想还是有区别的。

（1）从原则约束的侧重点来说，接口隔离原则更关注的是接口依赖程度的隔离，更加关注接口的“高内聚”；而单一职责原则更加注重的是接口职责的划分。

（2）从接口的细化程度来说，单一职责原则对接口的划分更加精细，而接口隔离原则注重的是相同功能的接口的隔离。接口隔离里面的最小接口有时可以是多个单一职责的公共接口。

（3）单一职责原则更加偏向对业务的约束，接口隔离原则更加偏向设计架构的约束。这个应该好理解，职责是根据业务功能来划分的，所以单一原则更加偏向业务；而接口隔离更多是为了“高内聚”，偏向架构的设计。

S: Single responsibility Principle (encapsulation, cohesion)

A class should have only one reason to change.

* Each class or entity should be responsible for one thing.
* Each should perform a set of closely related tasks.

O: Open Closed Principle (Abstraction, Polymorphism)

Class and functions should be open for extension but closed for modification.

* Open to extension:
  1. Can extend modules with new behavior
* Closed to modification:
  1. Behavior can be added without changing the source code of module

Liskov Substitution Principle (inheritance, Polymorphism)

Subclass should be substitutable for their base classes

* Classes should *correctly* fulfill any expected behaviors inherited from superclasses.

Interface Segregation Principle (Cohesion, Inheritance, Abstraction)

**Interfaces should be small, each dealing with one aspect of a problem.**

* “Fat”, general-purpose interfaces are not cohesive.
* An interface should allow only relevant behavior to be seen.
* A class should not be forced to implement irrelevant behavior

Dependency inversion (Abstraction)

**Classes should depend upon abstract concepts, rather than concrete implementations.**

* All modules should depend on high-level modules.
* Abstractions should not depend on details.
* Details should depend on abstractions.

OOD

Good design centers around the following goals:

* **Code reuse** 
  1. Can be reused for different data or applications
* **Scalability**
  1. Can handle increase in work or number of users
* **Extensibility**
  1. Easily extended or modified
* **Maintainability**
  1. Holds up to changing user needs

UML

What is UML?

Unified Modeling Language

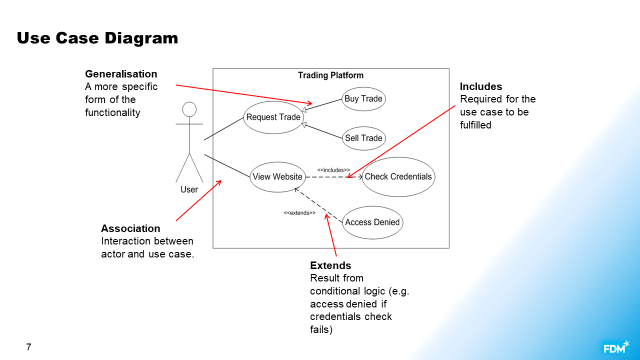
* Object oriented design
* Data structure modeling
* Scenario modeling
* Logical flow modeling

What are the two categories?

* Structural
* Behaviour

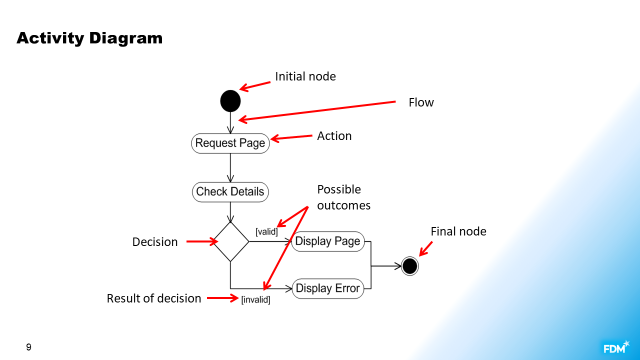
What is a Use Case diagram

A behavioral UML. High level view, from the perspective of outside the system. Models interaction between things external to the system (actors) and functionality within the system (use cases)



What is a Activity diagram

A low level view behavioral diagram. Models logical flow of the system.



What is a class diagram

Structural UML. Models the classes within an OOP program, including:

Their attributes and behaviors. Their relationships with other classes.

Generalization: is a type.

Realization: implements an interface

What are the different forms of dependency in a class diagram

Dependency: uses an object in methods

Association: has an object as a variable

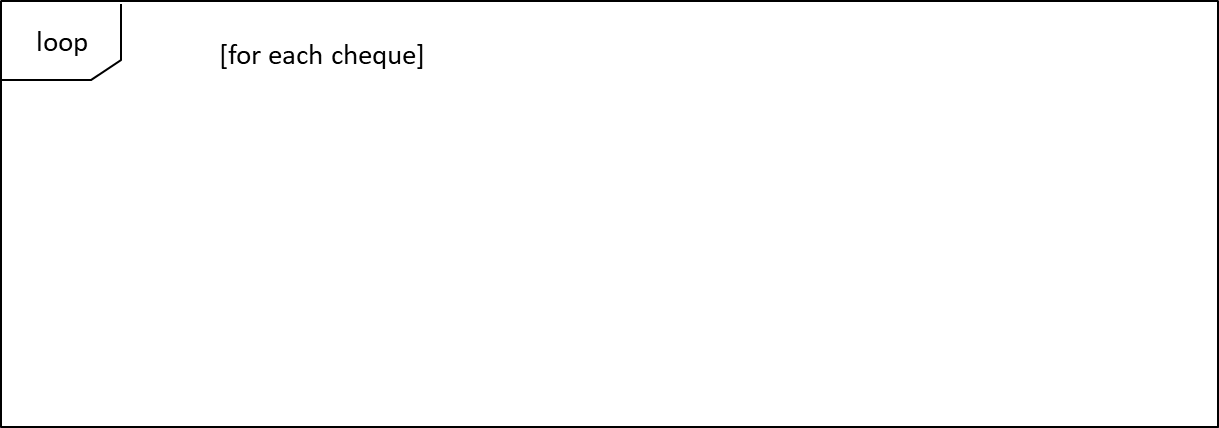
Aggregation: has zero or many

Composition: requires, has one or many

What is a sequence diagram

Behavioral UML. Models a single scenario for a single process. Shows the interactions that take place between classes involved in that process.

How do you represent if statements and loops in a sequence diagram?

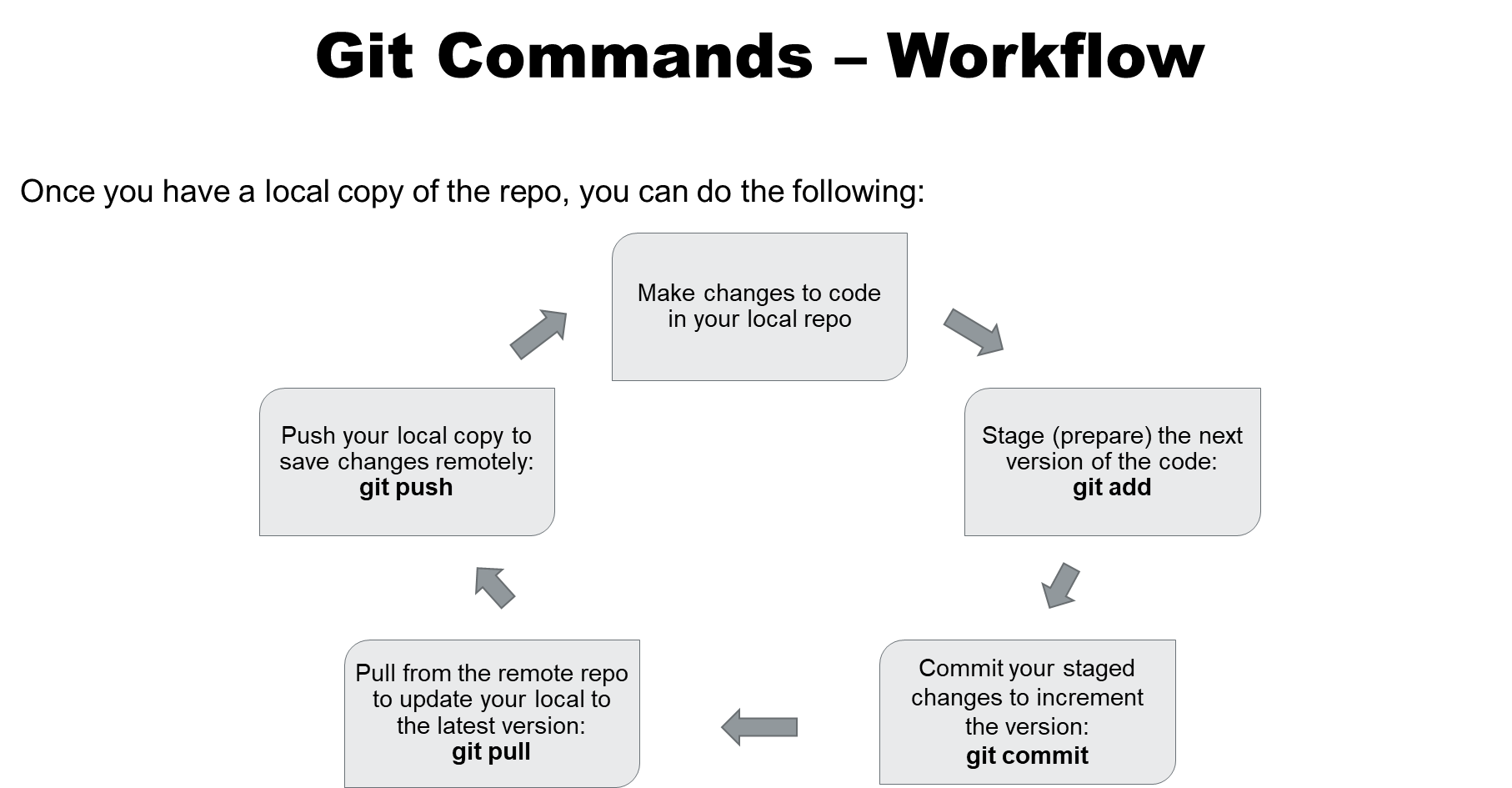


Source Control

What is version control?

**Version** **control** or **source control** refers to tracking   
and managing changes to information

In software development, this means changes to the code base.



What is the purpose of source control?

What tools are available?

What are some key best practices to use?

Name several Git commands and describe their purpose.