INTERNSHIP PRESENTATION

ANISHA VINEETH

SOFTWARE TRAINEE

SAYONE TECHNOLOGIES

OBJECT ORIENTED PROGRAMMING

- OOP is a style of programming that focuses on using objects to design and build applications.
- It allows users create the objects that they want and then create methods to handle those objects.
- Manipulating these objects to get results is the goal of Object Oriented Programming.
- OOP is a programming language model organised around objects rather than "actions" and data rather than logic.

Core OOPs concepts are ,

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

OOPS CONCEPTS:

OBJECT

- An object is a self-contained component that contains properties and methods needed to make a certain type of data useful.
- An object's properties are what it knows and its methods are what it can do.
- In Java, Objects can be initialised in any of the ways,
  - By reference variable
  - By method
  - By constructor

Initialising through reference:

```
class Student{
 int id;
 String name;
}
class TestStudent2{
 public static void main(String args[]){
  Student s1=new Student();
  s1.id=101;
  s1.name="Ann";
  System.out.println(s1.id+" "+s1.name);//printing members with a white space
 }
}
```

OOPS CONCEPTS:

OBJECT

Initialising through method:

- Here, We are creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method.
- And, we are displaying the state (data) of the objects by invoking the displayInformation() method.

```
class Student{
 int rollno;
 String name;
 void insertRecord(int r, String n){
  rollno=r;
  name=n;
 }
 void displayInformation(){System.out.println(rollno+" "+name);}
}
class TestStudent4{
 public static void main(String args[]){
  Student s1=new Student();
  Student s2=new Student();
  s1.insertRecord(111,"Karan");
  s2.insertRecord(222,"Aryan");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```

OOPS CONCEPTS:

- ## CLASS

  - A *class* is a blueprint or template or set of instructions to build a specific type of object.
  - Every object is built from a class.

```
//Java Program to demonstrate having the main method in another class
//Creating Student class.
class Student{
 int id;
 String name;
}
//Creating another class TestStudent1 which contains the main method
class TestStudent1{
 public static void main(String args[]){
  Student s1=new Student();
  System.out.println(s1.id);
  System.out.println(s1.name);
 }
}
```

OOPS CONCEPTS:

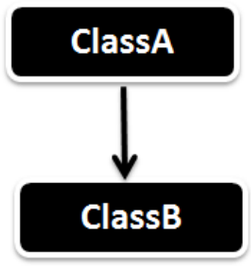
- ## INHERITANCE

  The process by which one class acquires the properties(data members) and functionalities(methods) of another class is called **inheritance**.

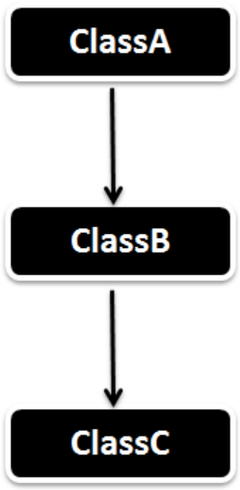
```
class Teacher{
        String destination = "Teacher";
        String collegeName = "CUSAT";
        void does(){
         System.out.println("Teaching");
        }
}
 public class OopsTeacher extends Teacher{
       String mainSubject = "OOPs";
      public static void main(String args[]{
      OopsTeacher obj = new OopsTeacher();
      System.out.println(obj.collegeName);
      System.out.println(obj.Destination);
      System.out.println(obj.mainSubject);
      obj.does();
       }
}
```

TYPES OF INHERITANCE:

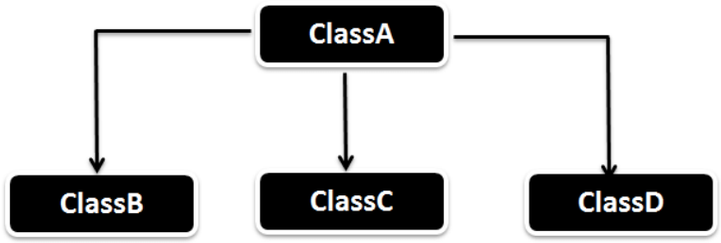
- **Single Inheritance**: refers to a child and parent class relationship where a class extends the another class.



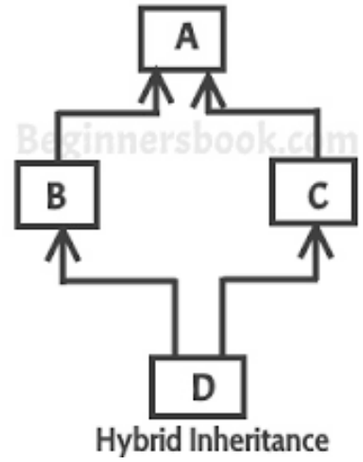
- **Multilevel inheritance**: refers to a child and parent class relationship where a class extends the child class. For example class C extends class B and class B extends class A.



- **Hierarchical inheritance**:

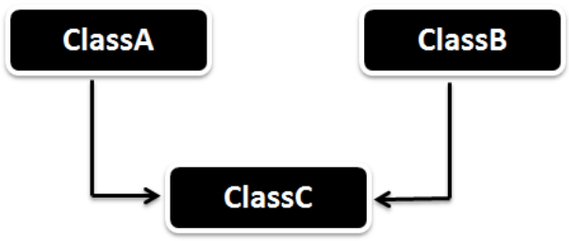


- **Hybrid inheritance**:



- **Multiple Inheritance**: refers to the concept of one class extending more than one classes i.e, a child class has two parent classes.

- *Java doesn't support multiple inheritance.*
- It doesn't allow multiple inheritance to **avoid the ambiguity** caused by it.
- Example of such problem is the **diamond problem** that occurs in multiple inheritance.
- However it can be implemented using Interface.



OOPS CONCEPTS:

POLYMORPHISM

- Polymorphism makes it possible to use the same entity in different forms.
- Java provides us with two ways to implement polymorphism: *method overloading* and *method overriding*.

```
class Bike{
  void run(){System.out.println("running");}
}
class Splendor extends Bike{
  void run()
{System.out.println("Running safely with 60km");}

  public static void main(String args[]){
    Bike b = new Splendor();
    b.run();
  }
}
```

Output:

Running safely with 60km.

OOPS CONCEPTS:

ABSTRACTION

- **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.
- Example of an abstract class that have abstract method:

```
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

OOPS CONCEPTS:

ENCAPSULATION

- **Encapsulation in Java** is a process of wrapping code and data together into a single unit.
- Example, a capsule which is mixed of several medicines.

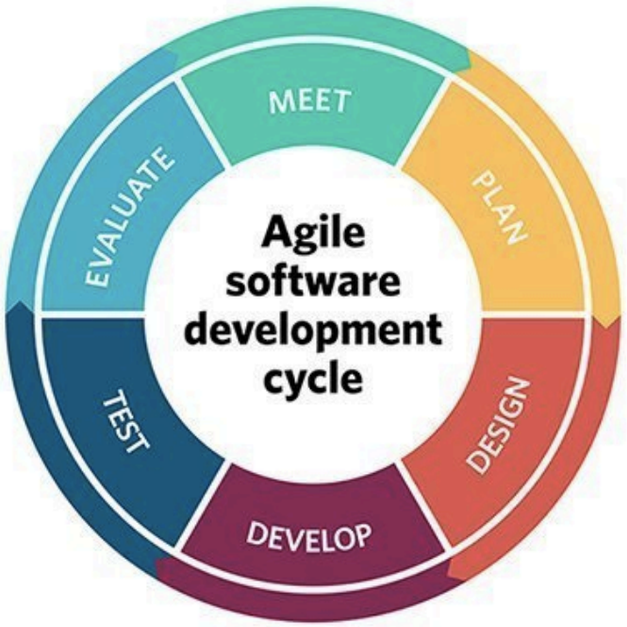
```
//A Java class which is a fully encapsulated class.
//It has a private data member and getter and setter methods.
public class Student{
//private data member
private String name;
//getter method for name
public String getName(){
return name;
}
//setter method for name
public void setName(String name){
this.name=name
}
}
```

POPULAR PROGRAMMING METHODOLOGIES:

- **Agile Software Development Model:**

- Agile is the ability to create and respond to change.
- The main goal of agile methods is minimising the risk by developing software in ***short time boxes, called iterations***, which typically last one to four weeks.
- Each time box is like a mini software project that includes all the tasks necessary to release the mini-increment of new functionality:
  - planning,
  - requirements analysis,
  - design,
  - coding,
  - testing, and
  - documentation.

Agile Software Development Model:



- Main principles of Agile software development methodology:
  - face-to-face meetings,
  - constant cooperation,
  - early and continuous delivery of the working software,
  - transparency.

Pros

- Adaptive approach that responds to changes favourably
- Allows for direct communication to maintain transparency
- Improved quality by finding and fixing defects quickly and identifying expectation mismatches early

Cons

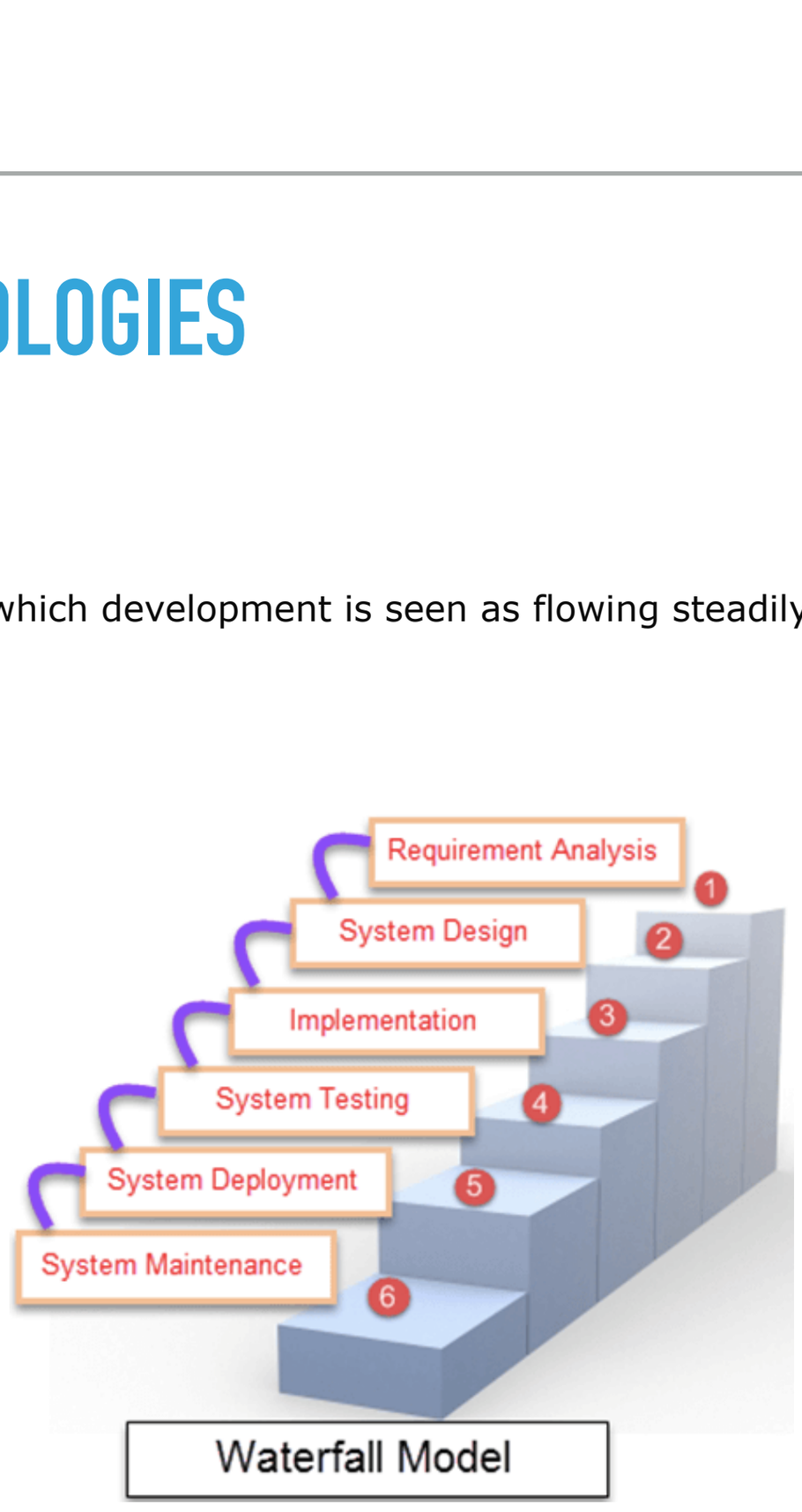
- Focuses on working with software and lacks documentation efficiency
- Chances of getting off-track as outcome are not clear

POPULAR PROGRAMMING METHODOLOGIES

- **Waterfall Model:**

The Waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through several phases, typically:

- analysis
- software requirements specification
- Software design
- testing
- integration (if there are multiple subsystems)
- deployment (or Installation)
- maintenance



Pros and cons of Waterfall Model:

Pros

- Easy to understand and functional
- Simple enough to handle as model is rigid
- Saves significant amount of time
- Allows for easy testing and analysis
- It allows for departmentalisation and managerial control

Cons

- Only matches precise needs
- Not applicable for maintenance projects
- Does not allow editing in the testing phase
- No option to know possible outcome of a project
- Not excellent for long and ongoing projects

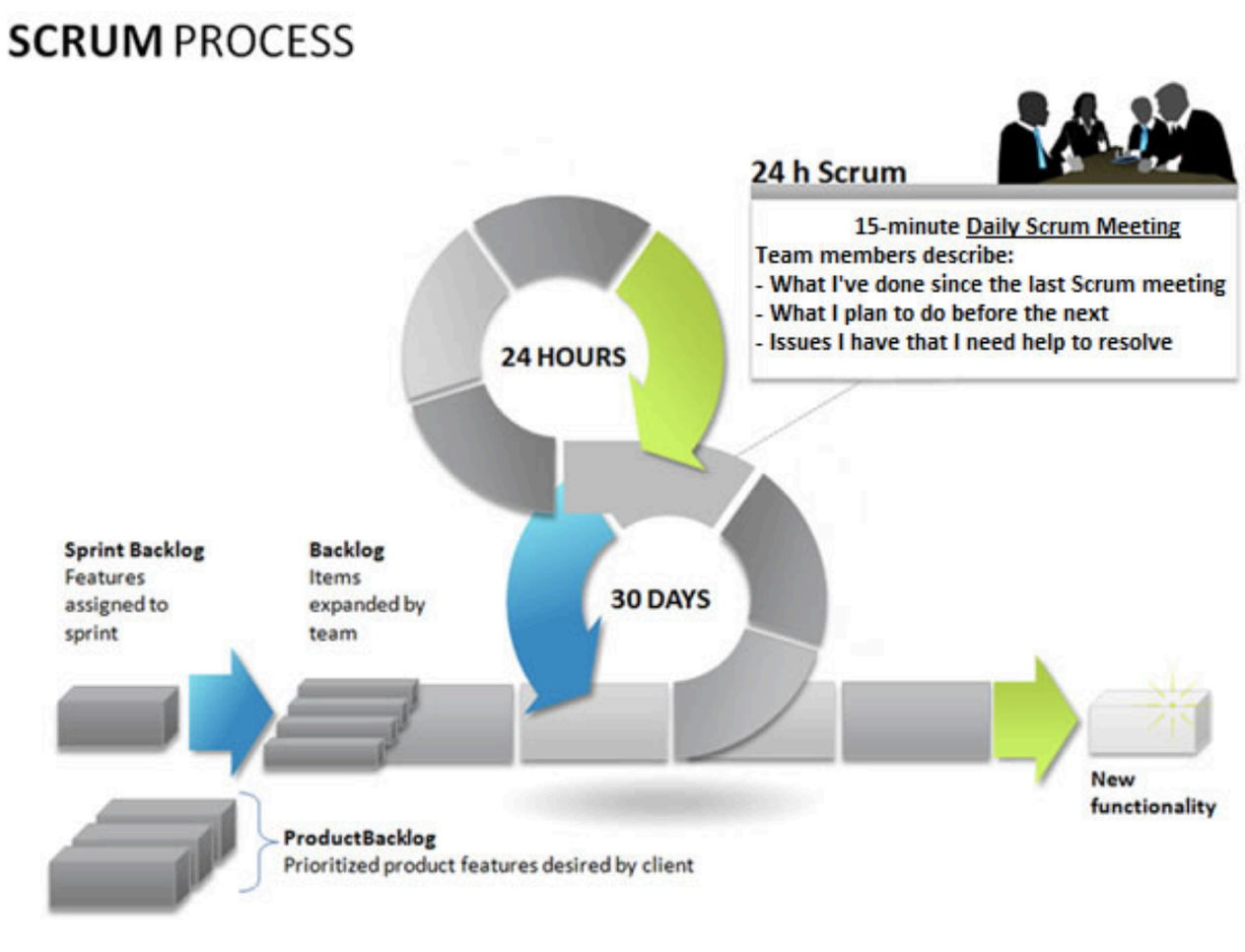
POPULAR PROGRAMMING METHODOLOGIES

- **Scrum Model:**

- Scrum is an agile way to manage a project, usually software development.

- It defines a flexible, holistic product development strategy where a development team works as a unit to reach a common goal.

- This method enables teams to self-organise by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved.



POPULAR PROGRAMMING METHODOLOGIES

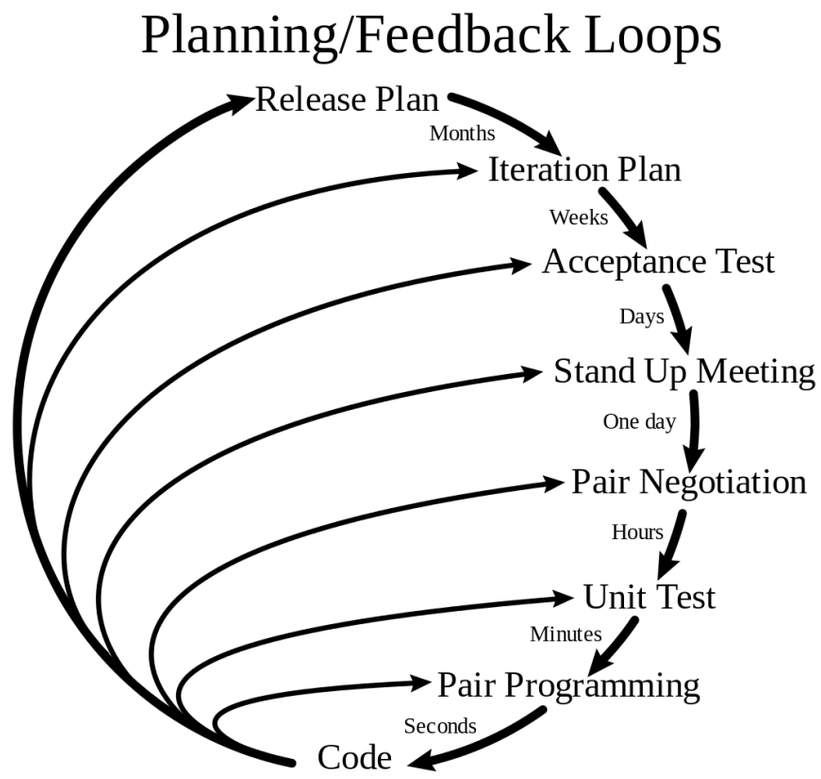
- **Extreme Programming:**

- refers to an agile software engineering methodology.
- It was created to avoid the development of functions that are not currently needed.
- Aimed at the creation of a top-notch final product with no regard for frequent changes in requirements.
- Another aim of this method is reducing the costs of software essentials.
- To achieve that, continuous testing and planning are applied.

Extreme Programming:

- In comparison with the other approaches, XP takes more time and human resources.

- It is the best choice if your client has a deadline to deliver the product with no clear understanding of how it must work, and the risk is higher.



Pros

- It lays focus on customer involvement
- Establishes rational plans and schedules

Cons

- Effectiveness depends on the people involved
- Requires frequent meeting for development raising total costs

PROGRAMMING METHODOLOGIES

- **Rapid Application Development Methodology:**

  - It aimed to reduce the amount of construction needed to build a product.
  - RAD is a condensed development process that produces a high-quality system with low investment costs.

The Rapid Application Development method is divided into four phases:

- requirements planning
- user design
- construction
- cutover

The user design and construction phases are repeated until the user approves that all of the requirements are met.

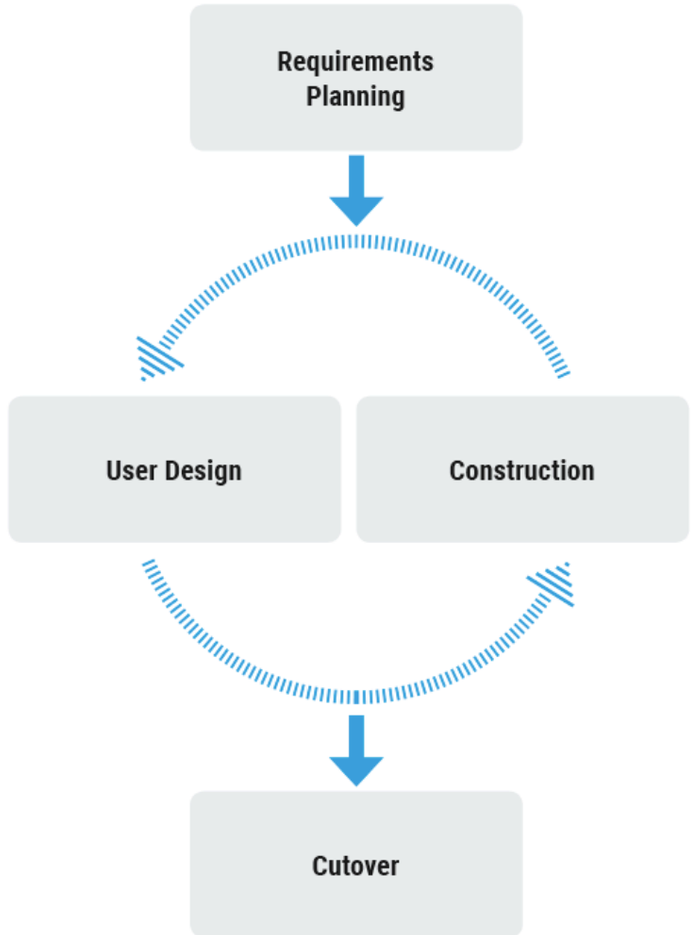
Rapid Application Development Methodology:

Pros

- Makes the entire development process effortless
- Assists client in taking quick reviews
- Encourages feedback from customers for improvement

Cons

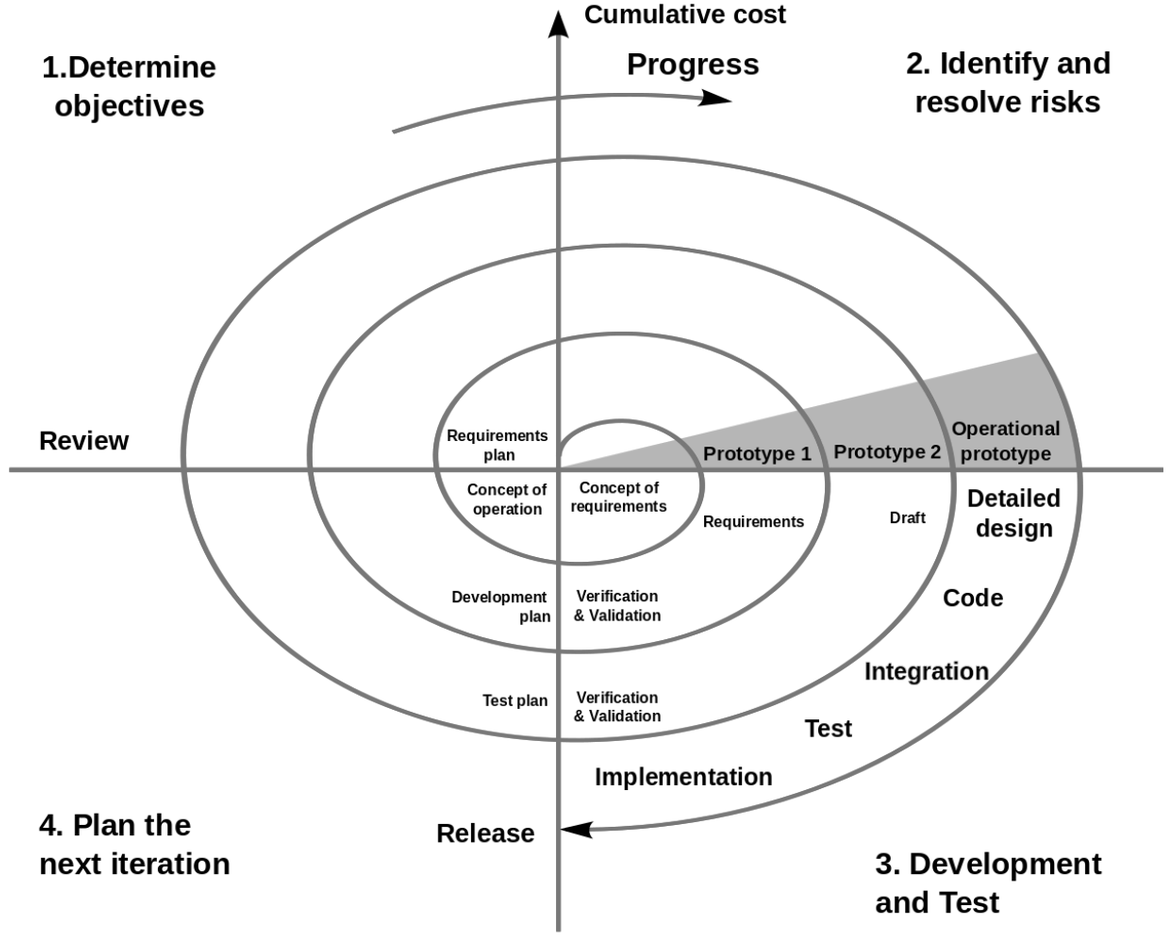
- Dependant on the team for performance
- Works on modularised system confined on this methodology
- Requires extremely skilled personnel to handle complexities
- Not applicable for the small budgeted projects



PROGRAMMING METHODOLOGIES

- **Spiral Model:**

  - The Spiral Lifecycle Model is a sophisticated lifecycle model that focuses on early identification and reduction of project risks.



Spiral Model:

- The Spiral methodology extends the Waterfall model by adding rapid prototyping in an effort to combine advantages of top-down and bottom-up concepts.

- A spiral project starts on a small scale, explores risks, makes a plan to handle the risks, and then decides whether to take the next step of the project.

- Success at using this Model depends on conscientious, attentive, and knowledgeable management.

Pros

- Risk factors are considerably reduced
- Excellent for large and complex projects

Cons

- Costly model in software development
- Failure in risk analysis phase may damage the whole project

PROGRAMMING LANGUAGES

- A set of commands, instructions, and other syntax used to create a software program.
- **High-level languages:** Languages that programmers use to write code .
- **Low-level language :** Code that is recognised directly by the computer hardware.
- Examples of high-level languages include C++, Java, Perl, and PHP.
- Low-level languages include assembly and machine languages.

PROGRAMMING LANGUAGES

- **Java:**
  - One of the most popular and widely used programming language and platform.
  - Key to its popularity : “write once, run anywhere” philosophy.
  - Java is fast, reliable and secure.
  - From desktop to web applications, scientific supercomputers to gaming consoles, cell phones to the Internet, Java is used in every nook and corner.

```java
public class AddTwoNumbers {

   public static void main(String[] args) {

      int num1 = 5, num2 = 15, sum;
      sum = num1 + num2;

      System.out.println("Sum of these numbers: "+sum);
   }
}
```

PROGRAMMING LANGUAGES

- **Objective C:**

  - A general-purpose, object-oriented programming language.
  - ***Use:*** used by Apple for the OS X and iOS operating systems and their respective APIs, Cocoa and Cocoa Touch.

```objc
#import <Foundation/Foundation.h>

int main (int argc, const char * argv[]) {
   NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];

   NSLog (@"hello world");
   [pool drain];
   return 0;
}
```

Objective C:

The **data types** in Objective C are:

- **Basic types**: integer types , floating point types
- **Enumerated types**
- **Type void**
- **Derived Types** : Pointer types, Array types, Structure types , Union types, Function types.

PROGRAMMING LANGUAGES

- **Swift:**

  - Swift 4 is a new programming language developed by Apple Inc for iOS and OS X development.

  - It’s a safe, fast, and interactive programming language

  - **Playground feature :** Swift 4 programmers can write their code in the playground and execute it to see the results immediately.

  - The first public release of Swift was released in 2010.

  - Basic Program:

```
import UIKit

var str = "Hello, World!"

print(str)
```

PROGRAMMING LANGUAGES

- **JavaScript:**
  - **JavaScript** (**JS**) is a lightweight, interpreted programming language.
  - Mostly used as client side scripting language.
  - Along with HTML and CSS, JavaScript is essential to front-end web development.
  - In addition to “pure” JavaScript, there are also a number of libraries and frameworks intended to make JavaScript development easier.
  - Some of the most popular frameworks include Angular, React, Vue , Ember and jQuery.

```
<!doctype html>
<html>
 <head>
    <script>
        alert('Hello world!');
    </script>
</head>
```

PROGRAMMING LANGUAGES

- **Python:**

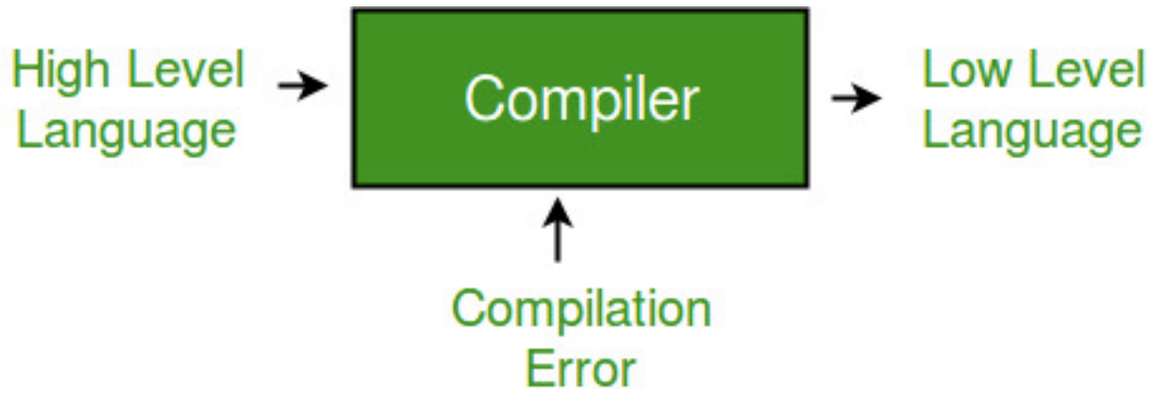
  - Python is an interpreted, interactive, object-oriented programming language.
  - It lets you work quickly and integrate systems more efficiently.
  - A high level programming language, allows you to focus on core functionality of the application by taking care of common programming tasks.

```
# Add two numbers
num1 = 3
num2 = 5
sum = num1+num2
print(sum)
```

  - Any line starting with # in Python programming is a comment.

COMPILER

- A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses.



Phases of a Compiler –
There are two major phases of compilation, which in turn have many parts. Each of them take input from the output of the previous level and work in a coordinated way.

Analysis Phase – An intermediate representation is created from the given source code

1. Lexical Analyzer
2. Syntax Analyzer
3. Semantic Analyzer

Synthesis Phase – Equivalent target program is created from the intermediate representation. It has three parts :

4. Intermediate Code Generator
5. Code Optimizer
6. Code Generator

SERVER SIDE:

- **Programming Languages:**

  - Writing any piece of code that runs on the server machine is server side programming.

  - Writing programs to create dynamic pages is ***server-side programming*** since the programs run on the web server.

  Programming Languages used for server side programming:

  - JavaScript
  - Python
  - Java

  Programming Languages used for client side programming:

  - JavaScript
  - HTML
  - CSS
  - C,C++

SERVER SIDE:

- **Frameworks:**

  - **Django**:

    - Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

    - Ridiculously fast.

    - Reassuringly secure.

    - Exceedingly scalable.

- Other frameworks include,

- **Flask**(Python)

- **Express**(Node js /javascript)

- **Laravel**(PHP)

REST API:

- RESTful API, an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

- It is based on representational state transfer (REST) technology, an architectural style and approach to communications often used in web services development.

- They use
  - -GET to retrieve a resource;
  - -PUT to change the state of or update a resource, which can be an object, file or block;
  - -POST to create that resource; and
  - -DELETE to remove it.

How RESTful APIs work

- A RESTful API breaks down a transaction to create a series of small modules.

- Each module addresses a particular underlying part of the transaction.

Web Pages:

- A Web page is a document for the World Wide Web that is identified by a unique uniform resource locator (URL).

Web Servers:

- A Web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computers' HTTP clients. Dedicated computers and appliances may be referred to as Web servers as well.

HISTORY OF MOBILE APPS:

- **What is mobile app?**
    - A mobile application, most commonly referred to as an app, is a type of application software designed to run on a mobile device, such as a smartphone or tablet computer. Mobile applications frequently serve to provide users with similar services to those accessed on PCs

- **History of mobile app development**
    - If you go back to the history of the mobile applications, then you can clearly figure out that a few Java games, a calculator or monthly calendar were all that came under the category of mobile apps.

    - The first smartphone was announced for the general use by IBM in 1993 that was equipped with the features like calculator, world clock, calendar and contact book.

    - The BlackBerry Smartphone released in 2002 was the next major achievement in the field of mobile application development and it was marked by BlackBerry Limited, formerly known as Research In Motion Limited (RIM) and integrated with the innovative concept of wireless email.

TYPES OF MOBILE APPS:

- **Native**
  - Developed for a single mobile operating system exclusively, therefore they are “native” for a particular platform or device.
  - App built for systems like iOS, Android, Windows phone, Symbian, Blackberry can not be used on a platform other than their own.
  - Main advantage is high performance and ensuring good user experience as developers use native device UI.
  - Some cons to native apps are higher cost compared to other types of apps – due to the need of creating app duplicates for other platforms, separate support and maintenance for different types of apps resulting in bigger product price.
  - There are mainly 3 areas of native apps
    - Android (Java/Kotlin)
    - iOS (Swift / Objective C)
    - Windows (.Net)

TYPES OF MOBILE APPS:

- **Hybrid**
  - Hybrid app is a web app hiding inside of a native wrapper — basically a tiny browser that only ever shows one web app.
  - They look and feel like a native app, but ultimately outside of the basic frame of the application
  - Frameworks for building hybrid apps are
    - Ionic
    - Cordova
    - Xamarin
    - PhoneGap etc..

TYPES OF MOBILE APPS:

- **Cross platform**

  - Cross-platform mobile app development is the process of creating mobile apps that can be deployed or published on multiple platforms using a single codebase, instead of having to develop the app multiple times using the respective native technologies for each platform.

  - Languages used for building cross platform apps are

    - React Native

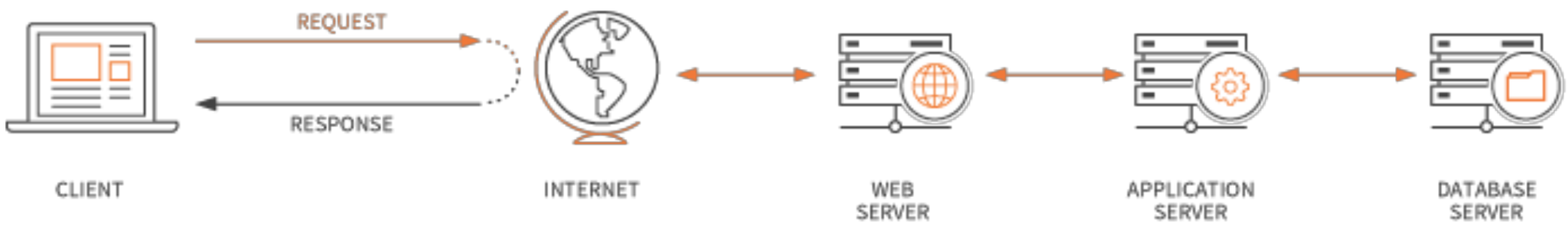
    - Flutter etc..

WORKING OF AN APP:

- **Web Apps:**

A typical web application flow looks like;

- **User** triggers a request to the **web server** over the **Internet**, either through a web browser or the application's user interface.
- Web server forwards this request to the appropriate web application server.
- Web application server performs the requested task – such as querying the database or processing the data – then generates the results of the requested data.
- Web application server sends results to the web server with the requested information or processed data.
- Web server responds back to the client with the requested information that then appears on the user's display



WORKING OF AN APP:

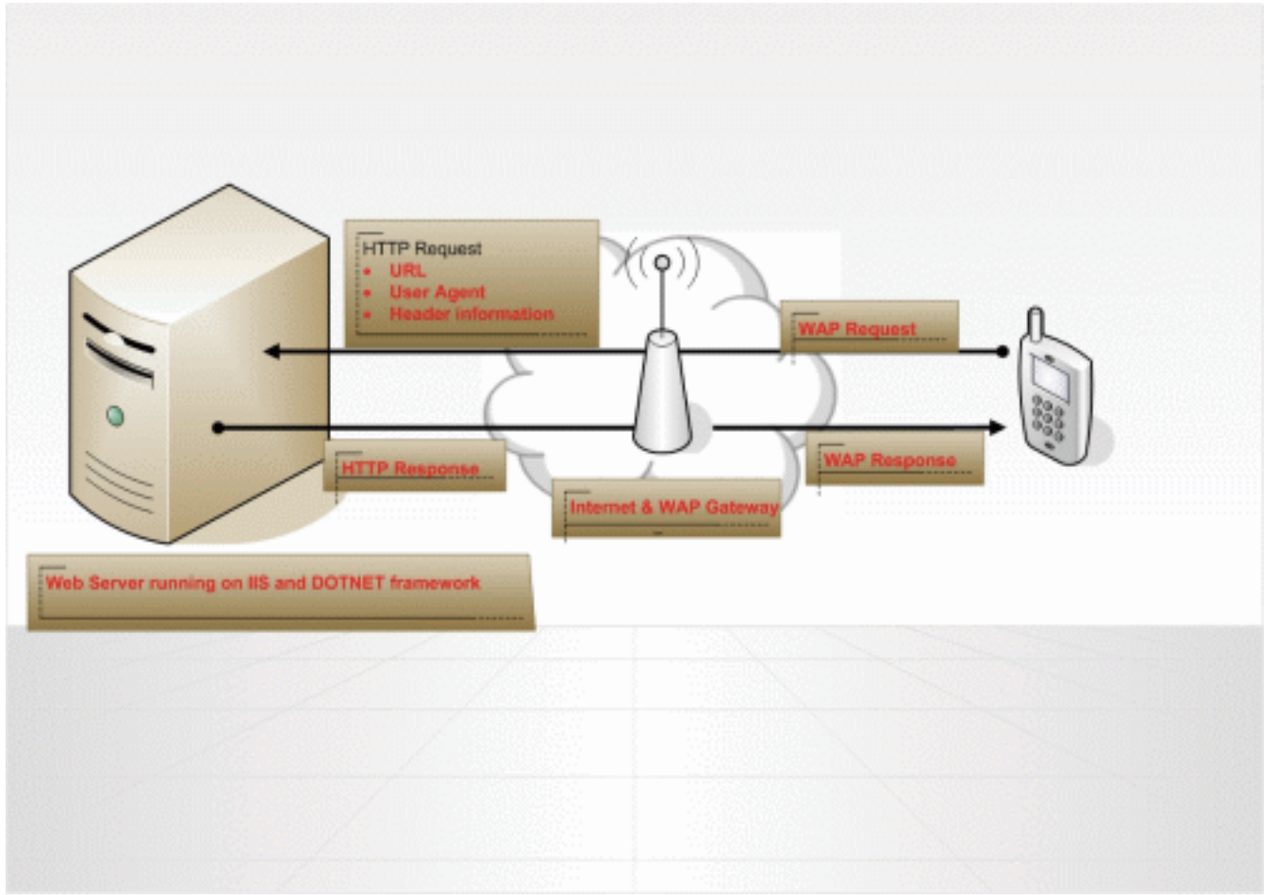
- **Mobile Apps:**

  - The HTTP request coming from a mobile device contains three major sections:

  - the URL,
  - User Agent and
  - Header information

  - The User Agent string contains the device info and this string will be used to machine.config to identify the details about the requesting mobile device.

  - The URL section contains the requested page.



DESIGN STANDARDS:

- Follow the basic principles to keep the design standards:
  - Design according to the platform
  - Maintain the design flow as per the theme
  - Less typing using autocomplete data
  - Make interface elements clearly visible
  - Content must be readable to user
  - Product must highlight while representing a portfolio
  - Define UI brand signatures
  - Use a performance dashboard
  - Optimize UI flows and elements
  - Make navigation self apparent

Thank You!