**1.** Proqlaşdırma paradiqmaları (paradigms) proqramlaşdırma dillərinin özündə saxladığı əsas xüsusiyyətlərdir. Bunu danışıq dilləri ilə müqayisə edərək misal göstərə bilərik. Məsələn, Azərbaycan dili, “altay” dillərinin “türk” dilləri qolundakı “oğuz” qrupuna aid olmaqla, özlüyündə 3 paradiqmaya məxsusdur. Deməli, “oğuz” dil qrupuna aid olan Azərbaycan və Türkmən dili qohum dillər sayılır. Bu məntiqi proqramlaşdırmaya tətbiq edərkən görürük ki, JAVA və C# dilləri özlüyündə “structured”, “object-oriented”, “imperative”, “generic”, “reflective” və “concurrent” paradiqma xüsusiyyətlərini daşıdığından qohum dillərdir.

Proqlaşdırmada mövcud olan paradiqmaların siyahısı :

Action

[Agent-oriented](https://en.wikipedia.org/wiki/Agent-oriented_programming)

[Array-oriented](https://en.wikipedia.org/wiki/Array_programming)

Aspect-oriented

[Automata-based](https://en.wikipedia.org/wiki/Automata-based_programming)

[Concurrent computing](https://en.wikipedia.org/wiki/Concurrent_computing)

[Data-driven](https://en.wikipedia.org/wiki/Data-driven_programming)

[Declarative](https://en.wikipedia.org/wiki/Declarative_programming)

Logic

[Constraint](https://en.wikipedia.org/wiki/Constraint_programming)

[Dataflow](https://en.wikipedia.org/wiki/Dataflow_programming)

[Dynamic/scripting](https://en.wikipedia.org/wiki/Dynamic_programming_language)

[Event-driven](https://en.wikipedia.org/wiki/Event-driven_programming)

Flow-driven

Functional

[Function-level](https://en.wikipedia.org/wiki/Function-level_programming)

[Generic](https://en.wikipedia.org/wiki/Generic_programming)

[Imperative](https://en.wikipedia.org/wiki/Imperative_programming)

[Literate](https://en.wikipedia.org/wiki/Literate_programming)

[Language-oriented](https://en.wikipedia.org/wiki/Language-oriented_programming)

[Metaprogramming](https://en.wikipedia.org/wiki/Metaprogramming)

[Non-structured](https://en.wikipedia.org/wiki/Non-structured_programming)

[Nondeterministic](https://en.wikipedia.org/wiki/Nondeterministic_programming)

Object-oriented

[Parallel computing](https://en.wikipedia.org/wiki/Parallel_computing)

[Probabilistic](https://en.wikipedia.org/wiki/Probabilistic_programming_language)

Procedural

Reflective

[Stack-based](https://en.wikipedia.org/wiki/Stack-oriented_programming)

[Structured](https://en.wikipedia.org/wiki/Structured_programming)

[Symbolic](https://en.wikipedia.org/wiki/Symbolic_programming)

[Value-level](https://en.wikipedia.org/wiki/Value-level_programming)

Quantum programming

Əsas 4-lük : imperative, declarative, functional, object-oriented.

Mənbə :

<https://en.wikipedia.org/wiki/Programming_paradigm>

<http://cs.lmu.edu/~ray/notes/paradigms>

**2.** “Imperative” və “declarative” paradiqmalar arasında 5 fərq :

I. “Imperative” paradiqmada əmrləri addım-addım detallı şəkildə yazırıq.

“Declarative”də isə ümumi olaraq əmrlərin məntiqinə əsaslanıb onları yığcam şəklə gətiririk.

II. “Imperative”də nəyi necə etdiyimiz önəmlidirsə, “declarative”də nə etdiyimiz əsasdır.

Məsələn, bir taksiyə minib sürücüyə hansı küçə və yollardan keçib ünvana çatmağımızı deyiriksə, bu hal “imperative” adlanır. Əgər taksi sürücüsünə yalnız gedəcəyimiz ünvanı bildiririksə, bu vəziyyət “declarative” hesab olunur.

III. “Imperative”də nəticənin hansı yolla əldə etdiyini bilmək olursa, “declarative”də yalnız nəticəni görmək olur.

IV. “Imperative”də developer kodları mürəkkəb hala gətirirsə, “declarative”də, əksinə, problemi həll etməyə köklənir.

V. “Imperative”də “if” və “then” kimi şərtlər mövcud olduğu halda “declarative”də yoxdur.

Mənbə :

<https://medium.com/front-end-weekly/imperative-versus-declarative-code-whats-the-difference-adc7dd6c8380>

<https://stackoverflow.com/questions/1784664/what-is-the-difference-between-declarative-and-imperative-programming/17403998#17403998>

<https://www.ukessays.com/essays/philosophy/the-imperative-programming-paradigm-philosophy-essay.php>

<http://www.himpub.com/documents/Chapter2220.pdf>

<https://searchitoperations.techtarget.com/definition/declarative-programming>

<https://en.wikipedia.org/wiki/Declarative_programming>

<https://en.wikipedia.org/wiki/Imperative_programming>

**3.** “Object Oriented Programming (OOP)” 4 hissədən ibarətdir :

Encapsulation, Abstraction, Inheritance, Polymorphism

Encapsulation – Bir obyektin malik olduğu statik və dinamik xüsusiyyətlər var. A obyektində olan xüsusiyyətlər B obyektindəki xüsusiyyətlərlə əlaqələli deyil, bir-biriləri üçün məxfidirlər.

Abstraction – Bir obyekt daxilində olan bütün xüsusiyyətlərin gizlədilməsidir.

Inheritance – “Child” (uşaq) A və B obyektlərinin malik olduğu xüsusiyyətlər “parent” (valideyn) X obyektinin ən azı bir xüsusiyyəti ilə eynidir. Yəni, onlar məlumatları irsi olaraq qəbul edir və eyni yerdə birləşir.

Polymorphism – A və B obyektləri bir neçə xüsusiyyətlərinə görə fərqlənsə də, eyni iş görmə xüsusiyyətləri də var.

Misallar :

I. “Samsung” və “I Phone” mobil telefonları

II. Rəqəmsal və Çoxfunksiyalı saat

III. Azərbaycan Manatı və ABŞ Dollar

I. Encapsulation - Hər ikisi müxtəlif sistem platformasında olmaqla öz spesifik xüsusiyyətlərinə malikdir.

Abstraction – Telefonlarda internetə qoşulmaq funksiyasını əmr edərkən biz prosesin necə getdiyini görmürük və yalnız internetə qoşulmaq əmrini vermiş oluruq. (Bu əmrin icrası zamanı blutuz aktiv olunmur, şəkil çəkilmir və s.)

Inheritance – Hər iki telefonun babası Qraham Bel’in icad etdiyi telefondur. Ona görə də zəng etmək funksiyaları eynidir.

Polymorphism (Mobil telefon – Canon fotoaparatı) – Hər ikisi şəkil çəkmək funksiyasına malikdir.

II. Encapsulation – Hər ikisində əqrəblər, rəqəmlər yerləşdirilsə də, quruluş, dizayn və funksiyaları cəhətdən fərqlənir.

Abstraction – Saatın neçə olduğunu bilirik, amma mexanizmin necə işlədiyini görmürük.

Inheritance – İlk ixtira edilən saatın, hətta nəticə etibarilə vaxtın müəyyənləşdirilməsini icra edən qum saatının işini görürlər.

Polymorphism (Saat - Kompyuter) – Hər ikisində vaxtı müəyyənləşdirmək mümkündür.

III. Encapsulation – Hər ikisinin xüsusi ölçüsü, rəngi, materialı və dəyəri var.

Abstraction – Bir məhsul alanda pulun ölçüsü, rəngi, materialı və dəyəri kimi xüsusiyyətlərinin əslində nə rol oynadığını anlamırıq.

Inheritance – Qədim zamanlarda istifadə olunan pul və sikkələr kimi mübadilədə istifadə olunur.

Polymorphism (Kağız pul - Kriptovalyuta) – Fiziki olaraq fərqlənsələr də, hər ikisi ilə ödəniş edib məhsul ala bilərik.

Mənbə :

<https://www.c-sharpcorner.com/UploadFile/cda5ba/object-oriented-programming-with-real-world-scenario/>

<https://dev.to/charanrajgolla/beginners-guide---object-oriented-programming>

<https://atomicobject.com/resources/oo-programming/the-oo-paradigm>

<https://medium.freecodecamp.org/object-oriented-programming-concepts-21bb035f7260>

<https://stackoverflow.com/questions/34730320/object-oriented-programming-paradigms>

**4.** Yığcamlığına görə “declarative” paradiqmə üstünlük verirəm. Hər şeyin detallı yazılması və səhv edərkən düzəltmək imkanının daha rahat olmasına görə isə “imperative” paradiqm əlverişlidir. Fikrimcə, bu paradiqmi bilmək digər paradiqmləri bilməyimizə asan yol açır. Həyatdakı misallara çox uyğun gəldiyindən mənasını anlamaq olur.

Mənbə : <https://en.wikipedia.org/wiki/Programming_paradigm>