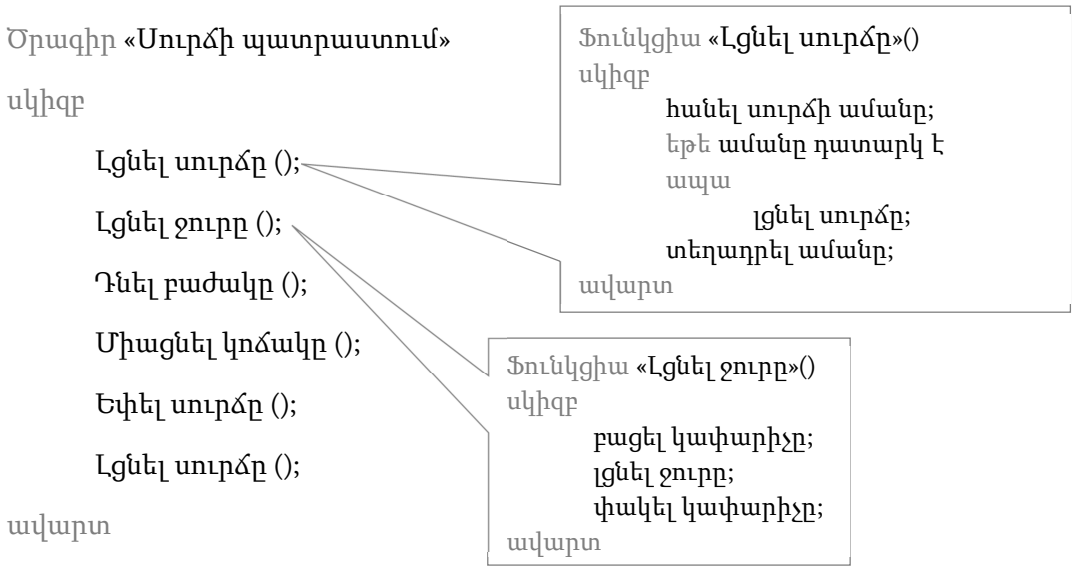
Օբյեկտով կողմնորոշված ծրագրավորում

Ի՞նչ է օբյեկտը և ինչո՞ւ է ստեղծվել ՕԿԾ-ն

Մինչ այժմ մենք դիտարկում էինք ընթացակարգային ծրագրավորման սկզբունքները: Ընթացակարգային ծրագրավորման հիմքում ընկած են ֆունկցիաները: Դա նշանակում է, որ կառուցվում են փոքր, ինքնուրույն ծրագրային կտորներ, և հիմնական ծրագիրը հավաքվում է այդ կտորներից:

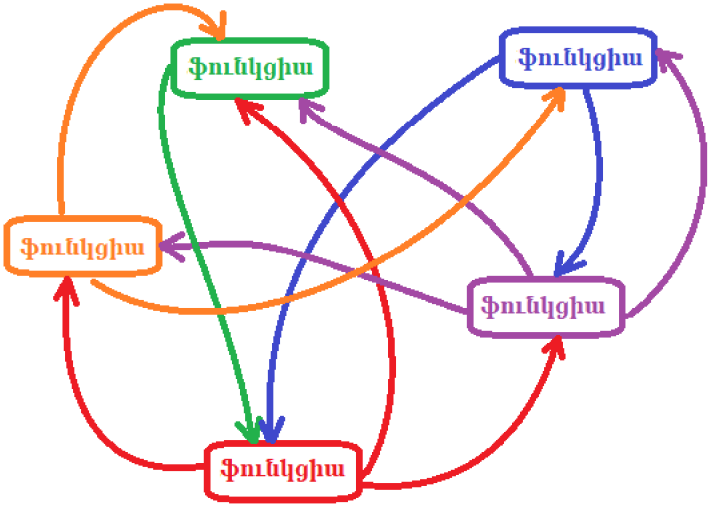
Օրինակ՝ դիտարկենք սուրճ պատրաստելու գործընթացը՝ կառուցված ընթացակարգային ծրագրավորման սկզբունքով:



Այս օրինակը ցույց է տալիս, թե ինչ հերթականությամբ են կատարվում գործողությունները սուրճ պատրաստելու համար: Ընդ որում, գործողությունները տրոհված են առանձին կտորների՝ ֆունկցիաների, որոնք հերթով կանչվում են հիմնական ծրագրից:

Ո՞րն է այս մոտեցման անհարմարությունը

Եթե մի փոքր ձևափոխենք մեր սուրճ պատրաստող սարքը և ավելացնենք շաքար լցնելու գործողությունը, ապա ստիպված կլինենք փոխել ամբողջ ծրագիրը, որպեսզի սարքը կարողանա պատրաստել նաև քաղցր սուրճ: Յուրաքանչյուր նոր գործողություն կամ գործողության փոփոխություն հանգեցնում է ծրագրի ամբողջական վերանայման և փոփոխության: Որքան շատանան նոր գործողությունները, այնքան կբարդանա ու կխճճվի ծրագիրը:



Ի՞նչ լուծում է առաջարկում ՕԿԾ-ն

Օբյեկտով կողմնորոշված ծրագրավորման (ՕԿԾ) գաղափարը առաջացել է այն պատճառով, որ փորձ է արվել իրական աշխարհի օբյեկտները արտապատկերել ծրագրավորման մեջ:

Օրինակ՝ դիտարկենք սուրճ պատրաստելու գործընթացը: Իրական կյանքում այս գործընթացում մասնակցում են երկու տարբեր օբյեկտներ՝ մարդը և սուրճ պատրաստող սարքը: Սուրճ պատրաստելու գործընթացում ամեն մի գործողությունը կատարում է օբյեկտներից որևէ մեկը:

Լցնել սուրճը – կատարում է մարդը
Լցնել ջուրը – կատարում է մարդը
Դնել բաժակը – կատարում է մարդը
Միացնել կոճակը – կատարում է մարդը
Եփել սուրճը – կատարում է սարքը
Լցնել սուրճը – կատարում է սարքը

Եթե իրական կյանքում գործում են երկու տարբեր օբյեկտներ, ապա ինչու ծրագրավորման մեջ ևս չդիտարկել երկու տարբեր օբյեկտներ՝ իրենց հատուկ գործողություններով:

Օրինակ.

Օբյեկտ «մարդ»	Օբյեկտ «սրճեփ»
գործողություններն են.	գործողություններն են.
լցնել սուրճը	եփել սուրճը
լցնել ջուրը	լցնել սուրճը
դնել բաժակը	
միացնել կոճակը	

Օբյեկտներով նկարագրելու դեպքում ծրագիրը զգալիորեն պարզանում է: Ծրագրի նկարագրությունը դառնում է ավելի հասկանալի և մոտ մարդու ընկալմանը:

Օրինակ՝ սուրճ պատրաստելու գործընթացը ՕԿԾ սկզբունքով կնկարագրվի այսպես.

```
Մարդ.նախապատրաստել ();
Մարդ. միացնել կոճակը  ();
Սրճեփ.պատաստել ();
```

Սա նշանակում է, որ մեր «մարդ» օբյեկտի համար կավելանա ևս մի գործողություն՝ «նախապատրաստել», որը կներառի սուրճը լցնելու, ջուրը լցնելու և բաժակը դնելու գործողությունները: Իսկ «սրճեփը» կունենա «պատրաստել» գործողությունը, որը կազմված կլինի սուրճը եփելու և լցնելու գործողություններից:

Օբյեկտ «մարդ»	Օբյեկտ «սրճեփ»
գործողություններն են.	գործողություններն են.
լցնել սուրճը ()	եփել սուրճը ()
լցնել ջուրը ()	լցնել սուրճը ()
դնել բաժակը ()	պատրաստել ()
միացնել կոճակը ()	
նախապատրաստել ()	

Եթե նաև քաղցր սուրճ պատրաստելու անհրաժեշտություն առաջանա, ապա «մարդ» օբյեկտի գործողություններին կավելացվի ևս մեկը «լցնել շաքարը» գործողությունը:

Օբյեկտ «մարդ» գործողություններն են.	Օբյեկտ «սրճեփ» գործողություններն են.
լցնել սուրճը () լցնել ջուրը () լցնել շաքարը () դնել բաժակը () միացնել կոճակը () նախապատրաստել ()	եփել սուրճը () լցնել սուրճը () պատրաստել ()

Յուրաքանչյուր նոր գործողություն ՕԿԾ-ում հանգեցնում է այդ գործողությունը կատարող օբյեկտի նկարագրության մեջ նոր ֆունկցիայի ավելացման: Ծրագրի ընդհանուր կառուցվածքը չի փոփոխվում:

Անկախ այն հանգամանքից՝ սուրճը պատրաստվում է քաղցր, թե դառը, հիմնական ծրագիրը նույնն է:

```
Մարդ.նախապատրաստել ();
Մարդ. միացնել կոճակը  ();
Սրճեփ.պատաստել ();
```

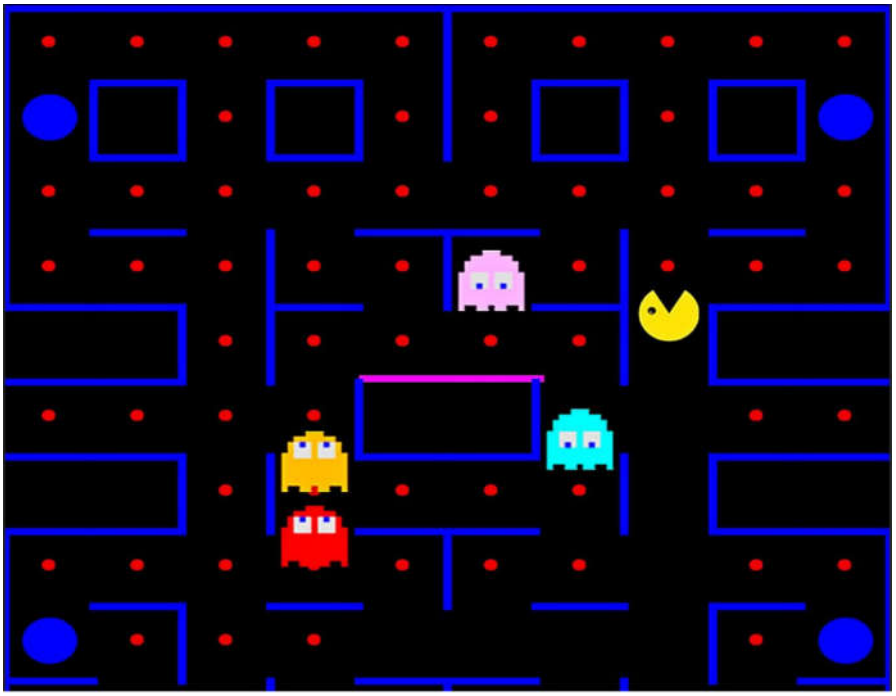
Ի՞նչ է օբյեկտով կողմնորոշված ծրագրավորումը

Օբյեկտով կողմնորոշված ծրագրավորումը (ՕԿԾ) մի մոտեցում է, երբ ամբողջ ծրագիրը կառուցվում է որպես փոխկապակցված օբյեկտների հավաքածու: Այն ստեղծվել է իրական կյանքի նմանությամբ: Իրական կյանքը ևս ներկայացնում է օբյեկտներ, որոնք փոխազդում և համագործակցում են միմյանց հետ:

Օրինակ՝ «ավտոմեքենա» օբյեկտը կառավարում է «վարորդ» օբյեկտը, որն իր հերթին առաջնորդվում է «լուսացույց» և «ճանապարհային երթևեկության նշան» օբյեկտներով:

ՕԿԾ-ում յուրաքանչյուր գործողություն կատարում է որևէ որոշակի օբյեկտ:

Օրինակ՝ դիտարկենք Pac-Man վիրտուալ խաղը: Դեղին «գնդիկը», շարժվելով լաբիրինթոսով, կուլ է տալիս կարմիր քարերը: Լաբիրինթոսով շրջում են նաև հրեշիկներ, որոնք վտանգավոր են «գնդիկի» համար:



Այստեղ յուրաքանչյուր կերպար հանդիսանում է օբյեկտ՝ գնդիկը, հրեշիկները: Օբյեկտ է նաև խաղադաշտը՝ լաբիրինթոսը: Այդ օբյեկտները փոխազդում են միմյանց հետ. հրեշիկը և գնդիկը կարող են շարժվել լաբիրինթոսով, հրեշիկը գնդիկին հանդիպելիս կարող է ոչնչացնել նրան: Գնդիկը կարող է կուլ տալ լաբիրինթոսում շաղ տված քարերը: Հրեշիկները միմյանց հանդիպելիս ոչինչ չեն անում:

Տվյալ խաղում կան հրեշիկներ, որոնք նման են միմյանց և՛ արտաքինով, և՛ իրենց հատուկ գործողություններով: ՕԿԾ-ում նմանատիպ օբյեկտները ներկայացվում են դասով (class):

Օրինակ.

1. Դիտարկենք յուրաքանչյուրիդ ձեռքում գտնվող գրիչը: Դրանք ևս օբյեկտներ են: Ձեր գրիչները տարբեր են իրենց գույնով, ձևով, օգտագործվածության աստիճանով: Բայց բոլոր գրիչները ծառայում են միևնույն նպատակին՝ նախատեսված են գրելու համար և ունեն ընդհանուր անվանում՝ «գրիչ»:
   Այսպիսով, կարող ենք բոլոր գրիչները համարել նույն «գրիչ» դասի օբյեկտներ: «Գրիչ» դասի օբյեկտը պետք է ունենա կաղապար, որը լցված է թանաքով, ունենա ծայր, որ ինչ-որ մակերևույթի հպելիս այդ ծայրից դուրս գա թանաքը:
   Բոլոր «գրիչ» դասին պատկանող օբյեկտները պարտադիր պետք է ունենան այդ դասի հատկությունները: Մյուս բոլոր հատկությունները տվյալ օբյեկտի առանձնահատկություններ են (գույն, ձև և այլն):
2. Դուք բոլորդ նստած եք աթոռներին: Ամեն աթոռ օբյեկտ է՝ իր առանձնահատկություններով: Բայց բոլոր աթոռներն ունեն ընդհանուր հատկություններ, որոնցով էլ պայմանավորված է «աթոռ» դասի նկարագիրը:
   «Աթոռ» դասի օբյեկտը պետք է ունենա ոտքեր, որոնց վրա դրված լինի հարթ մակերևույթ, և ունենա նստելու համար հարմար բարձրություն:




   Այս դասի բոլոր օբյեկտները պարտադիր ունեն այս դասի հատկությունները: Մյուս բոլոր հատկությունները կոնկրետ աթոռի (օբյեկտի) հատկություններ են՝ գույնը, ոտքերի քանակը, նյութը, որից պատրաստված է, թիկնակի առկայությունը և այլն:

Օբյեկտով կողմնորոշված ծրագրավորումը (ՕԿԾ) մի մոտեցում է (մեթոդ), որով ծրագիրը ներկայացվում է որպես օբյեկտների համախումբ, որում ամեն օբյեկտ պատկանում է որևէ դասի:

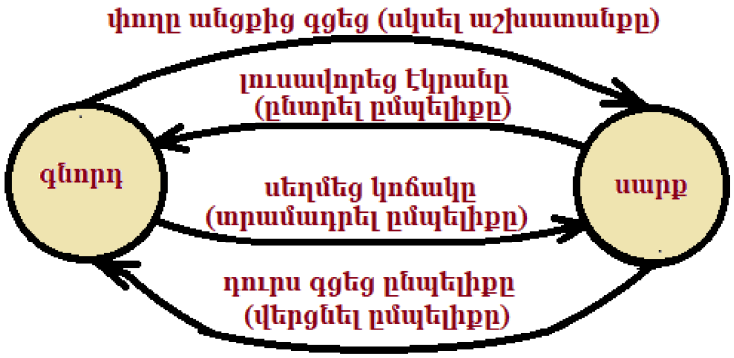
Փոխազդեցությունը օբյեկտների միջև

ՕԿԾ-ն ձևավորվել է՝ մոդելավորելով իրական կյանքը: Հետևաբար փոխազդեցությունը օբյեկտների միջև ՕԿԾ-ում պետք է կատարվի այնպես, ինչպես իրական կյանքում:

Օրինակ՝ դիտարկենք բոլորիս քաջածանոթ մի համակարգ՝ «ըմպելիք տրամադրող սարք»:

Ըմպելիքի գնման գործընթացին մասնակցում է երկու օբյեկտ՝ մարդը (գնորդը) և ըմպելիք տրամադրող սարքը:

Մոտենալով սարքին և փողը անցքից գցելով՝ գնորդը սարքին տալիս է ազդակ՝ «սկսել աշխատանքը»: Ստանալով այս ազդակը՝ սարքը էկրանին ցուցադրում է ըմպելիքների համարները և դրանով գնորդին տալիս է ազդակ «ընտրել ըմպելիքը»: Սեղմելով իր ցանկացած ըմպելիքի կոճակը՝ գնորդը սարքին հաղորդում է «տրամադրել ըմպելիքը» ազդակը: Դուրս գցելով ըմպելիքը՝ սարքը գնորդին հուշում է, որ ըմպելիքի գնման գործընթացն ավարտված է:

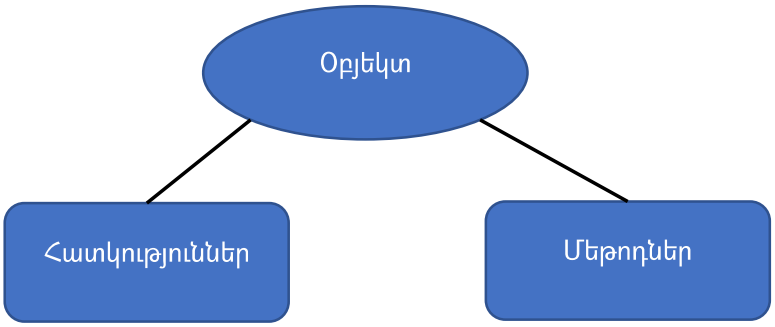


Օբյեկտները և նրանց հատկությունները

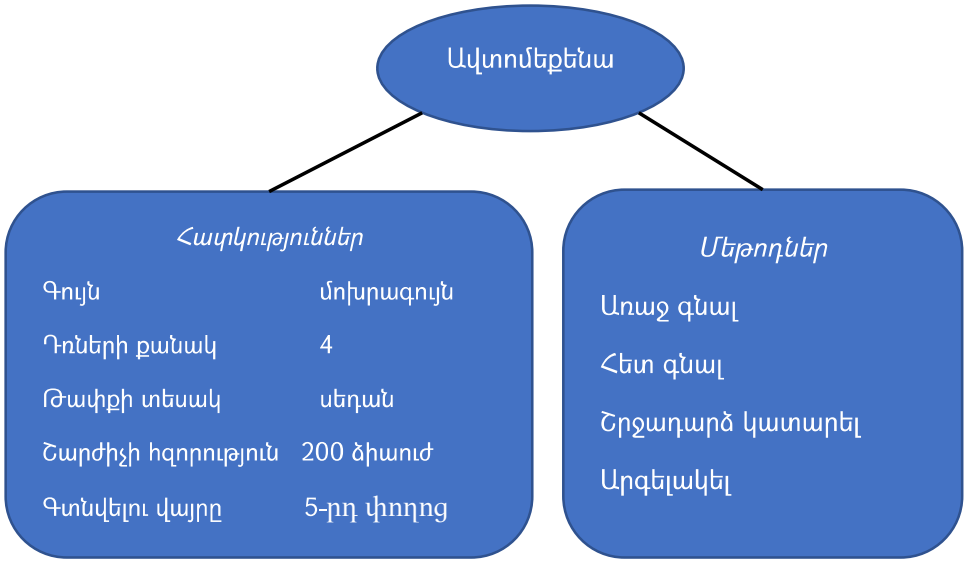
Ի՞նչ է օբյեկտը

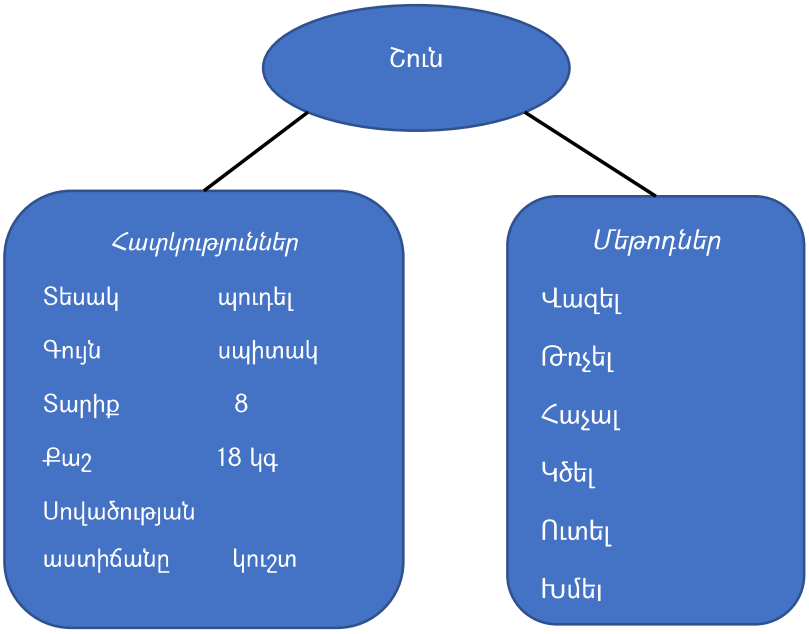
Օբյեկտն առանձին առարկա է, որն ունի հատկություններ և վարք:

Օբյեկտը կարելի է դիտարկել որպես որոշակի կառուցվածքով փոփոխական, որը պարունակում է ներկայացվող առարկայի կամ հասկացության մասին տվյալներ: Այդ տվյալները ցույց են տալիս թե՛ օբյեկտի **հատկանիշները**, թե՛ օբյեկտի գործողությունները: Այն գործողությունները, որոնք կարող է կատարել օբյեկտը, կոչվում են **մեթոդներ**:



Օրինակ





Օբյեկտն ամեն պահի կարող է գտնվել ինչ-որ վիճակում: Նրա վրա որևէ ազդեցություն կարող է փոփոխել օբյեկտի վիճակը:

Օրինակ.

«Ավտոմեքենա» օբյեկտը կարող է լինել անշարժ վիճակում: «Վարորդ» օբյեկտի կողմից ազդեցությամբ այն կարող է անշարժ վիճակից անցնել շարժվելու վիճակի, ինչի հետևանքով կփոխվի «ավտոմեքենա» օբյեկտի «գտնվելու վայրը» հատկությունը:

«Ավտոմեքենա» օբյեկտը մոխրագույն է, բայց «ավտոնորոգող» օբյեկտի ազդեցությունից հետո այն կարող է փոխել իր վիճակը և ձեռք բերել այլ հատկություն՝ կարմիր գույն:

Օբյեկտի տվյալ պահի վիճակը նկարագրվում է նրա տվյալ պահին ունեցած հատկություններով և տվյալ պահին նրա հնարավոր գործողություններով:

Օրինակ.

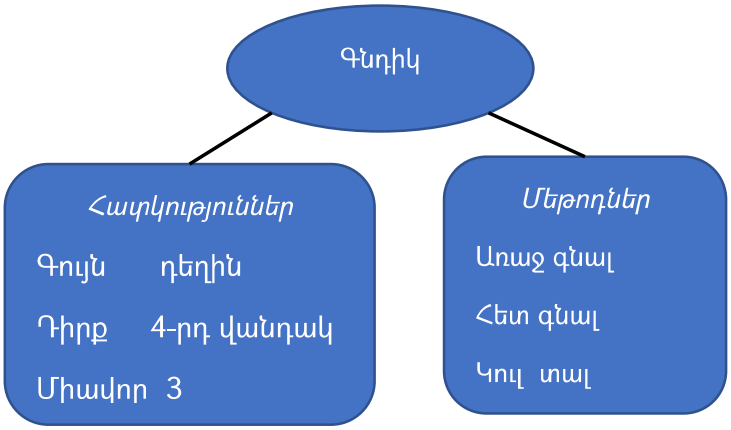
«Ավտոմեքենա» օբյեկտը ունի «արգելակել» մեթոդը: Բայց այդ մեթոդը կարող է կիրառվել միայն այն դեպքում, երբ ավտոմեքենան շարժման վիճակում է:

«Շուն» օբյեկտը կարող է «ուտել»: Բայց այս մեթոդը կիրառելի է միայն այն դեպքում, երբ շան առջև ուտելիք կա դրված: «Ուտելուց» հետո շան վիճակը փոխվում է՝ «սովածություն» հատկության «սոված» արժեքը դառնում է «կուշտ»:

Դասեր (classes)

ՕԿԾ-ի առավելություններից մեկն այն է, որ նույն կոդը պետք չէ մի քանի անգամ գրել: Օբյեկտը նկարագրվում է մեկ անգամ՝ իր հատկություններով և ֆունկցիաներով, և կարող է բազմակի օգտագործվել ծրագրում:

Օրինակ՝ Pac-Man խաղում դեղին գնդիկի խաղաքարերը կուլ տալու գործողությունը նկարագրվում է մեկ անգամ օբյեկտի մեթոդները նկարագրելիս: Խաղի ծրագրային կոդում ամեն անգամ, երբ գնդիկը պետք է խաղաքար կուլ տա, օգտագործվում է օբյեկտի «կուլ տալու» մեթոդը:



Իսկ ինչպե՞ս վարվել, երբ խաղին մասնակցում են նմանատիպ մի քանի օբյեկտներ:

Օրինակ՝ լաբիրինթոսով վազվզում են մի քանի գնդիկներ: Նրանք ներկայացվում են նույն պարամետրերով՝ անուն, գույն, դիրք, միավոր, և կարող են կատարել նույն գործողությունները՝ առաջ գնալ, շրջվել աջ, շրջվել ձախ, կուլ տալ:

Նույն մեթոդը (ֆունկցիան) մի քանի անգամ նկարագրելը օպտիմալ չէ: Նաև անիմաստ է: Ինչո՞ւ նույն ֆունկցիան կրկնել մի քանի անգամ: Այս խնդրի լուծումը դասերն են:

Դասը շաբլոն է օբյեկտի ստեղծման համար: Այն պարունակում է օբյեկտի հատկությունների և մեթոդների նկարագիրը:

Եթե համեմատենք դասական ծրագրավորման մեթոդների հետ, ապա դասը տիպն է, իսկ տվյալ դասի օբյեկտը՝ փոփոխականը:

```
// Դասի նկարագիրը C++ լեզվում
class դասի անվանումը
{
    // հատկությունների ցանկը
    // մեթոդների ցանկը
}
```

«Գնդիկ» դասի նկարագիրը C++ լեզվում:

```
class Gndik   // դասի անունը
{
        // հատկությունների ցանկը
   Color  col;    // գույնը
   int Position;  // դիրքը
      int point;   // միավորը

      // մեթոդների ցանկը
         void Go()  // նկարագրում է առաջ գնալու գործողությունը
   {
     Position++; // առաջ գնալիս հայտնվում է նոր վանդակում, որի համարը 1-ով մեծ է
                                    նախորդից
   }

    void GoBack()  // նկարագրում է հետ գնալու գործողությունը
   {
      Position--; // հետ գնալիս հայտնվում է նոր վանդակում, որի համարը 1-ով փոքր է
                                    նախորդից
   }

         void Swallow() // նկարագրում է խաղաքարը կուլ տալու գործողությունը
   {
            Point++;  // խաղաքարը կուլ տալիս գնդիկի մավորը ավելանում է 1-ով
    }
};
```

Կոդ 1

Խաղի ծրագրային կոդում օբյեկտը հայտարարելիս նշվում է, թե որ դասին է այն պատկանում: Խաղի ընթացքում գործողության անհրաժեշտության դեպքում տվյալ օբյեկտի համար նշվում է միայն դասի մեջ նկարագրված մեթոդի անվանումը:

```
void main()
{
  Gndik gndik1;            // հայտարարվում է Gndik տիպի gndik1 օբյեկտը
  gndik1.Col = Yellow;  //տրվում է gndik1 օբյեկտի սկզբնական վիճակը, գույնը
  gndik1.Position = 1;  // դիրքը,
  gndik1.point = 0;      // միավորը

  Gndik gndik2;  // հայտարարվում է Gndik տիպի gndik2 օբյեկտը
  Gndik2.Col = Red;      //տրվում է gndik2 օբյեկտի սկզբնական վիճակը, գույնը
  Gndik2.Position = 5; // դիրքը,
  Gndik2.point = 0;     // միավորը
    ...
    gndik2.Go();       // gndik2-ը շարժվում է առաջ
    gndik2.Swallow(); // gndik2-ը խաղաքար է կուլ տալիս
    ...
    Gndik1.GoBack();  // gndik1-ը հետ է գնում
    Gndik1.GoBack();  // gndik1-ը նորից է հետ գնում
    Gndik1.Swallow(); // gndik1-ը խաղաքար է կուլ տալիս
    ...
}
```

Կոդ 2

Դասեր (classes)

Գործնական աշխատանք

1. Նկարագրել «աշակերտ» դասը:
   - Ի՞նչ տվյալներ պիտի պարունակի դասը:
   - Ի՞նչ մեթոդներ (ֆունկցիաներ) պիտի ունենա դասը:

2. Նկարագրել «ուսուցիչ» դասը:
   - Ի՞նչ տվյալներ պիտի պարունակի դասը:
   - Ի՞նչ մեթոդներ (ֆունկցիաներ) պիտի ունենա դասը:

3. Նկարագրել դասը.
   num1 – իրական թիվ է
   num2 - իրական թիվ է
   sum(), subtraction(), multiplication(), division() կատարում են, համապատասխանաբար, գումարման, հանման, բազմապատկման և բաժանման գործողությունները num1-ի և num2-ի բոլոր թույլատրելի արժեքների համար:

4. Նկարագրել դասը.
   number – իրական թիվ է
   snum – ամբողջ թիվ է
   power() – մեթոդ է, որը հաշվում է number-ի snum աստիճանը միայն դրանց թույլատրելի արժեքների դեպքում:

5. Նկարագրել դասը.
   num1 – իրական թիվ է
   num2 – իրական թիվ է
   inRange() մեթոդը ստուգում է՝ արդյոք տրված թիվը պատկանո՞ւմ է [num1, num2] միջակայքին:
   inRange1() մեթոդը ստուգում է՝ արդյոք տրված թիվը պատկանո՞ւմ է (num1, num2) միջակայքին: