Τμήμα Ηλεκτρολόγων Μηχανικών
& Μηχανικών Υπολογιστών
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΛΟΠΟΝΝΗΣΟΥ

Μάθημα: **Τεχνικές Προγραμματισμού Υπολογιστών. (Εργαστηριακό μάθημα)**
Διδάσκων : Πεφάνης Ευάγγελος

2ο Project (*Θέματα Α έως ΣΤ*)
και το θέμα Z (advance) <u>το οποίο είναι απόλυτα προαιρετικό</u>

στην γλώσσα προγραμματισμού JavaScript.

<u>Θέμα Α.</u>

Κατασκευάστε ένα πρόγραμμα που να διαβάζει την βαθμολογία των Ν φοιτητών και να τυπώνει το αντίστοιχο αποτέλεσμα.

Το πρόγραμμα πρέπει να λειτουργεί ως εξής:

Ο χρήστης εισάγει στην αρχή των αριθμό των φοιτητών για τους οποίους θα περάσει την βαθμολογία.(Ν φοιτητές)

Μετά εισάγει τον βαθμό μαθήματος (*δεκαδικός αριθμός μεταξύ του 0.0 και του 10.0*) τον οποίο το πρόγραμμα θα πρέπει να χαρακτηρίζει ως εξής:

α) "Άριστα", εάν ο βαθμός είναι μεγαλύτερος ή ίσος του 8.50

β) "Λίαν Καλώς", εάν ο βαθμός είναι μικρότερος του 8.50 και μεγαλύτερος ή ίσος του 6.50

γ) "Καλώς", εάν ο βαθμός είναι μικρότερος του 6.50 και μεγαλύτερος ή ίσος του 5.00 και

δ) "Αποτυχία", εάν ο βαθμός είναι μικρότερος του 5.00.

Στο τέλος το πρόγραμμα θα πρέπει να εκτυπώνει το πλήθος των φοιτητών για κάθε περίπτωση α) έως δ).

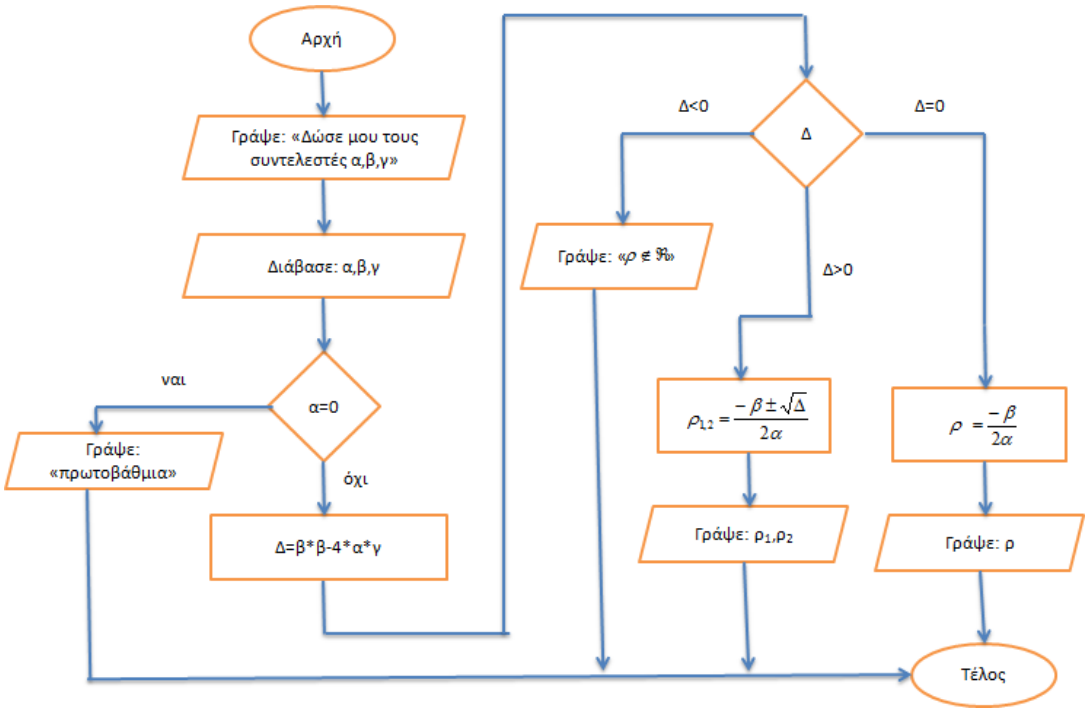
Οδηγίες.

Για την εισαγωγή των παραμέτρων όπου ζητούνται χρησιμοποιήστε την εντολή ***prompt().***
Τα αποτελέσματα του προγράμματος πρέπει να εκτυπώνονται στην **console** με τα ανάλογα σχόλια.

Για την υλοποίηση χρησιμοποιήστε την εφαρμογή JsEditor
που βρίσκετε στο Link : *https://jseditor.io/*

Θέμα Β.

Δίνεται το παρακάτω λογικό διάγραμμα επίλυσης της δευτεροβάθμιας εξίσωσης.



Κατασκευάστε ένα πρόγραμμα σε JavaScript που δέχεται σαν είσοδο ένα τριώνυμο:

$$f(x) = \alpha x^2 + \beta x + \gamma$$

το οποίο αντιστοιχεί στη λύση της δευτεροβάθμιας εξίσωσης:

$$\alpha x^2 + \beta x + \gamma = 0$$

Επιθυμούμε το πρόγραμμα που θα φτιάξουμε να αποφασίζει αν το τριώνυμο έχει, πόσες και ποιες πραγματικές ρίζες.
Στην περίπτωση που το τριώνυμο έχει πραγματικές ρίζες, τότε αυτές υπολογίζονται και τυπώνονται στην οθόνη.
Στην περίπτωση που το τριώνυμο δεν έχει πραγματικές ρίζες, τότε τυπώνεται ένα μήνυμα που μας ειδοποιεί ότι δεν υπάρχουν πραγματικές ρίζες.
Είσοδος στοιχείων απαιτείται κατά την εκκίνηση του προγράμματος όπου και ζητείται το τριώνυμο, οι συντελεστές **α**, **β** και **γ** δηλαδή.

Οδηγίες.
Για την εισαγωγή των παραμέτρων α, β και γ χρησιμοποιήστε την εντολή ***prompt().***
Πχ. Η παρακάτω εντολή παίρνει είσοδο τιμής από τον χρήστη και την καταχωρεί στην μεταβλητή b. ***Var b = prompt("Δώσε μου έναν αριθμό»);***

Τα αποτελέσματα του προγράμματος πρέπει να εκτυπώνονται στην **console** με τα ανάλογα σχόλια.
Μπορείτε να χρησιμοποιήσετε μεθόδους της βιβλιοθήκης **Math όπου είναι απαραίτητο.**

Για την υλοποίηση χρησιμοποιήστε την εφαρμογή JsEditor που βρίσκετε στο Link : https://jseditor.io/

Θέμα Γ.

Φτιάξτε ένα πρόγραμμα στο οποίο θα προσπαθούμε να μαντέψουμε έναν τυχαίο αριθμό από το 1 έως το 100.

Οδηγίες υλοποίησης της εφαρμογής.

- Όταν προσπαθείτε να μαντέψετε τον αριθμό, θα πρέπει το πρόγραμμα να σας κατευθύνει δίνοντας σας 2 πληροφορίες:

-Εάν ο αριθμός που δώσατε είναι μικρότερος από τον τυχαίο αριθμό, τότε θα εμφανίζεται το μήνυμα "Ο αριθμός που έδωσες είναι μικρότερος από αυτόν που σκέφτηκα".
-Εάν είναι μεγαλύτερος από τον τυχαίο αριθμό τότε θα εμφανίζεται το μήνυμα " Ο αριθμός που έδωσες είναι μεγαλύτερος από αυτόν που σκέφτηκα ".
-Εάν είναι ίσος με τον τυχαίο αριθμό τότε νικάτε στο παιχνίδι και εμφανίζεται αντίστοιχο μήνυμα.

- Όταν θα εμφανιστεί το μήνυμα ότι νικήσατε το παιχνίδι, τότε θα πρέπει να εμφανίζετε επίσης και πόσες προσπάθειες κάνατε μέχρι να μαντέψετε τον αριθμό.
-Για τη υλοποίηση του προγράμματος χρησιμοποιήστε μεταβλητές, συνθήκες και βρόγχους επανάληψης όπου είναι απαραίτητο.

Για την εισαγωγή του αριθμού που προσπαθείτε να μαντέψετε χρησιμοποιήστε την εντολή ***prompt().***

Για την εμφάνιση των αποτελεσμάτων και των μηνυμάτων χρησιμοποιήστε την εντολή **alert().**
Αυτή η εντολή εμφανίζει ένα παράθυρο στην οθόνη με τα μηνύματα που θέλουμε να δώσουμε στον χρήστη.
Π.χ : `alert("Ο αριθμός που είπες είναι μικρότερος από τον αριθμό που σκέφτηκα");`

Οδηγίες.

Για την εισαγωγή των παραμέτρων χρησιμοποιήστε την εντολή ***prompt().***
Πχ. Η παρακάτω εντολή παίρνει είσοδο τιμής από τον χρήστη και την καταχωρεί στην μεταβλητή b. ***Var b = prompt("Δώσε μου έναν αριθμό»);***

Τα αποτελέσματα του προγράμματος πρέπει να εκτυπώνονται στην **console** με τα ανάλογα σχόλια.
Τυχαίο αριθμό μπορούμε να παράγουμε με την χρήση των μεθόδων : Math.random() και Math.ceil(). Στην περίπτωσή μας δίνουμε τις παρακάτω εντολές:

```
var RND=100*Math.random();
var Random_Number=Math.ceil(RND);
```

Βοήθεια – Πληροφορίες – Παραδείγματα χρήσης των functions Math.random() , Math.ciel():
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/ceil

Για την υλοποίηση χρησιμοποιήστε την εφαρμογή JsEditor
που βρίσκετε στο Link : *https://jseditor.io/*

Θέμα Δ.

Χρήση των συναρτήσεων.

Να **γράψετε** δύο **συναρτήσεις**:

(1) η πρώτη συνάρτηση, να υπολογίζει *την περίμετρο της βάσης* ενός κυλίνδρου, όταν έχει σαν "είσοδό της" την ακτίνα.
Δίνεται ο τύπος: **Περίμετρος = 2*π*ακτίνα** (όπου το π είναι 3.1415)

(2) η δεύτερη συνάρτηση, να υπολογίζει *τον όγκο* ενός κυλίνδρου, όταν έχει σαν "εισόδους της" την ακτίνα και το ύψος.
Δίνεται ο τύπος: **Όγκος = π*ακτίνα^2*ύψος** (όπου το π είναι 3.1415)

Να γράψετε ένα πρόγραμμα που:

α) Να ζητά από τον χρήστη την **ακτίνα** της βάσης και το **ύψος** ενός κυλίνδρου και ακολούθως να τα αναθέτει σε μεταβλητές.

β) Να **καλεί** τις **συναρτήσεις** που φτιάξατε, για να υπολογίσει την περίμετρο και τον όγκο.

γ) Τα αποτελέσματα κάθε συνάρτησης, δηλαδή η περίμετρος και ο όγκος, να τυπωθούν στην console με ανάλογα μηνύματα.

Το πρόγραμμα πρέπει υποχρεωτικά να υλοποιηθεί με τρεις διαφορετικούς μεθόδους

Α μέθοδος : functions με παραμέτρους

Β μέθοδος : functions με καθολικές (Global) μεταβλητές

Οδηγίες.

Για την εισαγωγή των παραμέτρων χρησιμοποιήστε την εντολή ***prompt().***
Πχ. Η παρακάτω εντολή παίρνει είσοδο τιμής από τον χρήστη και την καταχωρεί στην μεταβλητή b.
Var b = prompt("Δώσε μου έναν αριθμό»);
Τα αποτελέσματα του προγράμματος πρέπει να εκτυπώνονται στην **console** με τα ανάλογα σχόλια.

Για την υλοποίηση χρησιμοποιήστε την εφαρμογή JsEditor
που βρίσκετε στο Link : https://jseditor.io/

Θέμα Ε.

Χρήση των συναρτήσεων.

Κατασκευάστε έναν απλό υπολογιστή (calculator) σε JavaScript

Κατασκευάστε ένα πρόγραμμα που θα εκτελεί τις βασικές αριθμητικές πράξεις (+,-,*,/) μεταξύ δύο **πραγματικών αριθμών**.
Καθώς επίσης και τις πράξεις **ύψωση σε δύναμη** και **τετραγωνική ρίζα** ενός αριθμού.
Μπορείτε εάν θέλετε να εμπλουτίσετε την εφαρμογή και με περισσότερους υπολογισμούς, (π.χ υπολογισμός ημιτόνου, συνημίτονου κ.α)

Α) Για την υλοποίηση των πράξεων πρέπει να χρησιμοποιηθούν απαραίτητα συναρτήσεις.

Β) **Να δημιουργηθεί επίσης συνάρτηση που να περιέχει τον κώδικα του προγράμματος (*από το Θέμα Β : υπολογισμός ριζών τριωνύμου*) και να ενσωματωθεί ως επιλογή μαζί με τις άλλες functions του calculator.**

Απαραίτητα να υπάρχουν σχόλια στον κώδικα και ο κώδικας να τηρεί όλους τους κανόνες σύνταξης.

Οδηγίες.

Να ελέγξετε την εκτέλεση του προγράμματος με ένα παράδειγμα.

Για την υλοποίηση χρησιμοποιήστε την εφαρμογή JsEditor
που βρίσκετε στο Link : https://jseditor.io/

Θέμα ΣΤ.

Χρήση των αντικειμένων και των μεθόδων τους.

Φτιάξτε ένα αντικείμενο **κύκλος** που θα έχει τις παρακάτω ιδιότητες:

α) Τις **συντεταγμένες** του κέντρου του κύκλου.
β) Την **ακτίνα** του κύκλου.

Θα έχει επίσης τις μεθόδους **Διάμετρος**, **Περίμετρος** και **Εμβαδόν** που θα υπολογίζουν την διάμετρο την περίμετρο και το εμβαδόν του κύκλου

Στην συνέχεια να γράψετε ένα πρόγραμμα που:

α) Να ζητά από τον χρήστη την **ακτίνα** του κύκλου και τις **συντεταγμένες του κέντρου του** και ακολούθως να τα αναθέτει στις ιδιότητες (**properties**) του αντικειμένου.

β) Να **καλεί** τις **μεθόδους** που φτιάξατε, για να υπολογίσει την διάμετρο την περίμετρο και το εμβαδόν.

γ) Τα αποτελέσματα εκτέλεσης της κάθε μεθόδου, να τυπωθούν στην console με ανάλογα μηνύματα.

Οδηγίες.

Για την εισαγωγή των παραμέτρων χρησιμοποιήστε την εντολή ***prompt().***
Πχ. Η παρακάτω εντολή παίρνει είσοδο τιμής από τον χρήστη και την καταχωρεί στην μεταβλητή b.
Var b = prompt("Δώσε μου έναν αριθμό»);
Τα αποτελέσματα του προγράμματος πρέπει να εκτυπώνονται στην **console** με τα ανάλογα σχόλια.

Το πρόγραμμα πρέπει να υλοποιηθεί με τρεις διαφορετικούς τρόπους (*οι β και γ είναι προαιρετικοί*)

α - με αντικείμενο κύκλος χωρίς κρυφές ιδιότητες

β - με αντικείμενο κύκλος με κρυφές ιδιότητες και μεθόδους Getters and Setters (**προαιρετικό**)

γ - με κλάση κύκλος με κρυφές ιδιότητες και μεθόδους Getters and Setters (**προαιρετικό**)

Για την υλοποίηση χρησιμοποιήστε την εφαρμογή JsEditor
που βρίσκετε στο Link : https://jseditor.io/

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

(Για την επίλυση των ασκήσεων συμβουλευτείτε τις σημειώσεις εργαστηρίου και διαλέξεων θεωρίας του μαθήματος στο eclass καθώς και το παράρτημα για το ΣΤ θέμα)

Θέμα Z : (απόλυτα προαιρετικό)

Όσοι φοιτητές/τριες επιθυμούν αφού μελετήσουν το Παράρτημα που ακολουθεί ας προσπαθήσουν να υλοποιήσουν το παρακάτω πρόγραμμα.
Όποιοι το ολοκληρώσουν θα έχουν την δυνατότητα να το παρουσιάσουν και να εξηγήσουν την λειτουργία του στο εργαστήριο.

Κάθε φύλλο (**Card**) μιας τράπουλας χαρακτηρίζεται από την όψη του (face) (Άσσος, Δύο, Τρία, …, Δέκα, Βαλές, Ντάμα, Ρήγας) και από το είδος της (suit), σύμβολο : (Κούπα, Σπαθί, Καρό, Μπαστούνι).

Μία τράπουλα(**Deck**) αποτελείτε από 52 φύλλα, 13 από κάθε είδος με τιμές από ένα έως δέκα και τις τρεις φιγούρες. Κάθε τράπουλα μπορεί να ανακατευθεί με τυχαία σειρά (μέθοδος shuffle) και μπορεί να επιστρέψει (μέθοδος deal) το φύλλο που βρίσκεται στην κορυφή της.

Να δημιουργηθούν οι παραπάνω κλάσεις και πρόγραμμα σε JavaScript που δημιουργεί μια τράπουλα, να την ανακατεύει και να τυπώνει τα φύλλα μετά το ανακάτεμα και πριν από αυτό.

Τα προγράμματα της εργασίας πρέπει να ανέβουν στο σύστημα εργασιών του eclass σε ένα αρχείο .txt με τίτλο **Project_Lab_JavaScript.txt**

ΠΑΡΑΡΤΗΜΑ. (αφορά τους προαιρετικούς τρόπους επίλυσης β) και γ) του θέματος ΣΤ καθώς και το Θέμα Ζ)

Βοήθημα : Επιπλέον στοιχεία (*με παραδείγματα*) για τον αντικειμενοστραφή προγραμματισμό στην JavaScript.

Βασικές Έννοιες

1. Αντικείμενα (Objects)

Τι είναι ένα αντικείμενο;

- Ένα αντικείμενο είναι μια συλλογή από **ζεύγη κλειδιών-τιμών** (key-value pairs).
- Τα **κλειδιά** είναι συμβολοσειρές (strings) ή `Symbols`, και οι **τιμές** μπορούν να είναι οποιοδήποτε τύπος δεδομένων (π.χ., αριθμοί, συμβολοσειρές, συναρτήσεις, ακόμα και άλλα αντικείμενα).

Χαρακτηριστικά Αντικειμένων

1. **Δυναμική Δομή:**
   - Τα αντικείμενα στη JavaScript είναι ευέλικτα. Μπορείτε να προσθέτετε ή να διαγράφετε ιδιότητες οποιαδήποτε στιγμή.
   - Παράδειγμα:

```
const car = { brand: "Toyota" };
car.model = "Corolla"; // Προσθήκη ιδιότητας
delete car.brand;      // Διαγραφή ιδιότητας
```

2. **Πρωτότυπα (Prototypes):**
   - Κάθε αντικείμενο στη JavaScript έχει ένα **πρωτότυπο (prototype)**, το οποίο είναι ένα άλλο αντικείμενο από το οποίο κληρονομεί ιδιότητες και μεθόδους.
   - Παράδειγμα:

```
const obj = {};
console.log(obj.toString()); // Η μέθοδος `toString` προέρχεται
από το Object.prototype
```

3. **Πρόσβαση σε Ιδιότητες:**
   - **Δια μέσου σημειογραφίας τελείας**: `object.property`.
   - **Δια μέσου σημειογραφίας αγκύλων**: `object["property"]`.

- Παράδειγμα:

```
const person = { name: "Alice", age: 25 };
console.log(person.name);       // Output: Alice
console.log(person["age"]);    // Output: 25
```

4. **Μέθοδοι (Methods):**
   - Οι μέθοδοι είναι ιδιότητες που περιέχουν συναρτήσεις.
   - Παράδειγμα:

```
const person = {
  name: "Alice",

  greet() {        //μέθοδος
    console.log(`Hello, I am ${this.name}`);
  }
};
person.greet(); // Output: Hello, I am Alice
```

---

2. Κλάσεις (Classes)

Τι είναι μια κλάση;

- Μια κλάση είναι ένα πρότυπο για τη δημιουργία αντικειμένων.
- Εισήχθη στην έκδοση ES6 της JavaScript για να προσφέρει μια πιο οργανωμένη και φιλική προς τον προγραμματιστή σύνταξη για αντικειμενοστραφή προγραμματισμό.

Σύνταξη μιας κλάσης

```
class ClassName {
  constructor() {
    // Αρχικοποίηση ιδιοτήτων
  }

  // Μέθοδοι
  methodName() {
    // Κώδικας
  }
}
```

Κύρια Στοιχεία της Κλάσης

1. **Constructor:**
   - Μια ειδική μέθοδος που καλείται αυτόματα όταν δημιουργείται ένα αντικείμενο από την κλάση.
   - Χρησιμοποιείται για την αρχικοποίηση ιδιοτήτων.
   - Παράδειγμα:

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
}

const person = new Person("Alice", 25);
```

```
console.log(person.name); // Output: Alice
```

2. **Μεθόδοι:**
   - Ορίζουν τη συμπεριφορά της κλάσης.
   - Παράδειγμα:

```
class Dog {
  constructor(name) {
    this.name = name;
  }

  bark() {
    console.log(`${this.name} says Woof!`);
  }
}

const dog = new Dog("Buddy");
dog.bark(); // Output: Buddy says Woof!
```

3. **Static Methods:**
   - Ανήκουν στην κλάση και όχι στα αντικείμενα της.
   - Παράδειγμα:

```
class MathUtils {
  static add(a, b) {
    return a + b;
  }
}

console.log(MathUtils.add(2, 3)); // Output: 5
```

---

3. Κρυφές Ιδιότητες (Private Properties)

Τι είναι οι κρυφές ιδιότητες;

- Οι κρυφές ιδιότητες είναι ιδιότητες που δεν είναι άμεσα προσβάσιμες από εξωτερικό κώδικα.
- Αυτές προσφέρουν **ενθυλάκωση (encapsulation)** και προστατεύουν τα δεδομένα από ακούσιες αλλαγές.

Μέθοδοι Υλοποίησης Κρυφών Ιδιοτήτων:

1. **Χρήση Προθέματος _:**
   - Συμβολική ιδιωτικότητα. Δεν αποτρέπει την πρόσβαση αλλά υποδεικνύει στον προγραμματιστή ότι η ιδιότητα είναι "ιδιωτική".
   - Παράδειγμα:

```
javascript
Αντιγραφή κώδικα
class BankAccount {
  constructor(initialBalance) {
    this._balance = initialBalance; // "Ιδιωτική" ιδιότητα
  }

  deposit(amount) {
    if (amount > 0) {
```

```
      this._balance += amount;
    }
  }

  getBalance() {
    return this._balance;
  }
}

const account = new BankAccount(100);
console.log(account.getBalance()); // Output: 100
```

2. **Χρήση του #:**
   - Πραγματική ιδιωτικότητα, εισήχθη στο πρότυπο ES2022 της JavaScript.
   - Παράδειγμα:

```
class BankAccount {
  #balance; // Κρυφή ιδιότητα

  constructor(initialBalance) {
    this.#balance = initialBalance;
  }

  deposit(amount) {
    if (amount > 0) this.#balance += amount;
  }

  getBalance() {
    return this.#balance;
  }
}

const account = new BankAccount(100);
console.log(account.getBalance()); // Output: 100

/* Η παρακάτω εντολή θα εκτελεστεί με σφάλμα γιατί η ιδιότητα
είναι κρυφή και πρέπει να κληθεί μόνο μέσω της μεθόδου
getBalance() */

console.log(account.#balance); // Error: Private field '#balance'
must be accessed inside the class
```

Πλεονεκτήματα της χρήσης Κρυφών Ιδιοτήτων

1. **Ασφάλεια Δεδομένων:**
   - Τα δεδομένα προστατεύονται από ακούσιες τροποποιήσεις.
2. **Ελεγχόμενη Πρόσβαση:**
   - Η πρόσβαση στις ιδιότητες γίνεται μέσω `getters` και `setters`, διασφαλίζοντας την εγκυρότητα των δεδομένων.
3. **Απλοποίηση του Κώδικα:**
   - Οι κρυφές ιδιότητες διαχωρίζουν την εσωτερική λογική από τη δημόσια διεπαφή.

Αναλυτικότερη παρουσίαση των πλεονεκτημάτων της χρήσης Κρυφών Ιδιοτήτων

Οι κρυφές ιδιότητες (private properties) είναι ένα βασικό εργαλείο για τη δημιουργία ασφαλών και καλά οργανωμένων εφαρμογών στον αντικειμενοστραφή προγραμματισμό. Ακολουθούν τα πλεονεκτήματα της χρήσης τους:

---

1. Ασφάλεια Δεδομένων

- **Προστασία της εσωτερικής κατάστασης ενός αντικειμένου:**
  - Οι κρυφές ιδιότητες είναι μη προσβάσιμες απευθείας από εξωτερικό κώδικα, αποτρέποντας τις ακούσιες ή σκόπιμες τροποποιήσεις.
  - Παράδειγμα:

```
class BankAccount {
  #balance; // Κρυφή ιδιότητα

  constructor(initialBalance) {
    this.#balance = initialBalance;
  }

  deposit(amount) {
    if (amount > 0) this.#balance += amount;
  }

  getBalance() {
    return this.#balance;
  }
}

const account = new BankAccount(100);
console.log(account.#balance); // Error: Cannot access private
field
```

- **Αποτροπή παραβίασης των κανόνων του αντικειμένου:**
  - Οι εξωτερικοί χρήστες δεν μπορούν να εισάγουν μη έγκυρα δεδομένα απευθείας στις κρυφές ιδιότητες, εξασφαλίζοντας τη συνέπεια.

---

2. Ελεγχόμενη Πρόσβαση μέσω Getters και Setters

- **Αυστηροί κανόνες πρόσβασης:**
  - Η πρόσβαση στις κρυφές ιδιότητες γίνεται μέσω `getters` και `setters`, όπου μπορεί να εφαρμοστεί πρόσθετος έλεγχος.
  - Παράδειγμα:

```
class Temperature {
  #celsius;

  constructor(celsius) {
    this.#celsius = celsius;
  }

  get fahrenheit() {
```

```
    return (this.#celsius * 9) / 5 + 32;
  }

  set celsius(value) {
    if (value >= -273.15) this.#celsius = value; // Θερμοκρασία
μεγαλύτερη από το απόλυτο μηδέν
    else console.log("Invalid temperature!");
  }
}

const temp = new Temperature(25);
console.log(temp.fahrenheit); // Output: 77
temp.celsius = -500; // Output: Invalid temperature!
```

- **Ευκολία επέκτασης:**
  - Οι `getters` και `setters` επιτρέπουν την εύκολη προσθήκη νέας λογικής χωρίς να τροποποιήσετε την κρυφή ιδιότητα.

---

3. Μειωμένη Πολυπλοκότητα και Καλύτερη Οργάνωση

- **Απόκρυψη Εσωτερικής Λογικής:**
  - Ο εξωτερικός χρήστης βλέπει μόνο τη δημόσια διεπαφή (`public interface`) και δεν χρειάζεται να κατανοήσει τις εσωτερικές λεπτομέρειες.
  - Παράδειγμα:

```
class Counter {
  #count = 0;

  increment() {
    this.#count++;
  }

  get value() {
    return this.#count;
  }
}

const counter = new Counter();
counter.increment();
console.log(counter.value); // Output: 1
```

- **Διαχωρισμός ευθυνών:**
  - Η κρυφή λογική επικεντρώνεται στη διαχείριση των δεδομένων, ενώ η δημόσια διεπαφή επικεντρώνεται στη λειτουργικότητα.

---

4. Ανθεκτικότητα σε Μελλοντικές Αλλαγές

- **Διατήρηση Συμβατότητας:**
  - Η εσωτερική λογική μπορεί να αλλάξει χωρίς να επηρεαστούν τα εξωτερικά μέρη του κώδικα που χρησιμοποιούν την κλάση.
  - Παράδειγμα:

```
class Database {
  #connection;
```

```
  connect() {
    if (!this.#connection) {
      this.#connection = "Connected to database"; // Μπορεί να
αλλάξει εσωτερικά
    }
    return this.#connection;
  }
}

const db = new Database();
console.log(db.connect()); // Output: Connected to database
```

---

5. Προστασία από Κακόβουλη Χρήση

- **Αποτροπή αλλαγών από εξωτερικό κώδικα:**
  - Η χρήση κρυφών ιδιοτήτων αποτρέπει τρίτους από το να "σπάσουν" την κλάση ή να παρακάμψουν τη λογική της.
- **Ασφάλεια σε πολύπλοκες εφαρμογές:**
  - Σε μεγάλα έργα, όπου πολλές ομάδες εργάζονται ταυτόχρονα, οι κρυφές ιδιότητες προστατεύουν τον κώδικα από ανεπιθύμητες αλλαγές από άλλους προγραμματιστές.

---

6. Εξοικονόμηση Πόρων

- **Μειωμένο debugging:**
  - Με τις κρυφές ιδιότητες, τα bugs (σφάλματα) που προκύπτουν από ανεξέλεγκτες τροποποιήσεις των δεδομένων μειώνονται δραματικά.
- **Αύξηση απόδοσης:**
  - Τα αντικείμενα με καθορισμένες δημόσιες διεπαφές είναι πιο εύκολα στη συντήρηση και ανάπτυξη.

Συμπέρασμα

Η χρήση κρυφών ιδιοτήτων είναι απαραίτητη για τη δημιουργία ασφαλούς, καθαρής και επεκτάσιμης αρχιτεκτονικής κώδικα. Προσφέρει ασφάλεια, καλύτερη οργάνωση, ανθεκτικότητα στις αλλαγές και προστασία από κακόβουλη ή απρόσεκτη χρήση. Ειδικά σε σύνθετες εφαρμογές, τα οφέλη τους είναι ανεκτίμητα.