128. Περιγράψτε τη χρησιμότητα των μεταβλητών κατά τη δημιουργία παιχνιδιών σε κώδικα JAVA (JAVA Game Variables).

Κατα τη δημιουργία ενος παιχνιού σε κώδικα JAVA, χρησιμοποιούμε τις JAVA Game Variables για να «ξέρουμε» ποια είναι η τρέχουσα κατάσταση του παιχνιδιού. Αυτό μπορεί να περιλαμβάνει το σκορ του παίκτη, τον αριθμό των εχθρικών διαστημικών πλοίων που υπάρχουν ή το που βρίσκονται και τι κάνουν όλα τα αντικείμενα του παιχνιδιού στην οθόνη.

129. Να γραφεί απλό πρόγραμμα με κώδικα σε Java, που να τυπώνει ως αποτέλεσμα στον χρήστη τις λέξεις "I love JAVA Game Programming"

```
JFrame frame = new JFrame("I love JAVA Game Programming");
JPanel panel = new JPanel();
JLabel label = new JLabel("I love JAVA Game Programming");
panel.add(label);
frame.add(panel);
frame.setVisible(true);
```

130. Να αναφέρετε τη χρησιμότητα των μεθόδων (methods) στο JAVA Game Programming.

Οι χρησιμοποίηση Java μεθόδων στην ανάπτυξη JAVA Game λογισμικού κρίνεται σκόπιμη για τους εξής λόγους:

Επαναχρησιμοποιήσιμο κώδικα
Εάν πρέπει να κάνουμε το ίδιο πράγμα, ή σχεδόν το ίδιο πράγμα, πολλές φορές, γράφουμε μια μέθοδο για να το κάνει, και στη συνέχεια απλά καλούμε τη μέθοδο κάθε φορά που πρέπει να εκτελεστεί αυτή η εργασία.

Παραμετροποίηση τον κώδικα
Εκτός από τον επαναχρησιμοποιήσιμο κώδικα που είναι ο ίδιος σε όλες τις περιπτώσεις, θα θέλαμε συχνά να χρησιμοποιήσουμε παραμέτρους που αλλάζουν τον τρόπο με τον οποίο λειτουργεί η μέθοδος.

Top-Down programming
Ένα πολύ χρήσιμο στυλ προγραμματισμού ονομάζεται "top-down" προγραμματισμός. Αντιμετωπίζουμε ένα μεγάλο πρόβλημα (το "κορυφαίο"), διασπώντας το σε μικρά προβλήματα. Για να το κάνετε αυτό σε ένα πρόγραμμα, γράφετε μια μέθοδο για την επίλυση του μεγάλου προβλήματός σας καλώντας άλλες μεθόδους για να λύσετε τα μικρότερα τμήματα του προβλήματος (Διαίρει και Βασίλευε). Οι μέθοδοι για την επίλυση των απλούστερων προβλημάτων παρομοίως απαιτούν άλλες μεθόδους μέχρι να καταλήξετε σε απλές μεθόδους που λύουν απλά προβλήματα.

Δημιουργία Εννοιολογικών Μονάδων

Δημιουργούμε μεθόδους για να κάνουμε κάτι που είναι μια ενέργεια/διεργασία η οποία ενοιολογικά επιλύει ένα τμήμα του προβλήματος, συνθέτοντας τμήμα της λύσης του.

Απλοποίηση
Επειδή οι τοπικές μεταβλητές και δηλώσεις μιας μεθόδου δεν μπορούν να φανούν εκτός της μεθόδου, αυτές (και η πολυπλοκότητά τους) κρύβονται από άλλα μέρη του προγράμματος, γεγονός που αποτρέπει τυχαία σφάλματα ή σύγχυση.

131. Ένα παιχνίδι που έχει δημιουργηθεί σε JAVA, συνήθως περιέχει:1) Frame2) Canvas3) GameLoop4) Rendering method5) Update method6) Mouse and Key input Εξηγείστε συνοπτικά τις έννοιες που αναφέρθηκαν, καθώς και τη χρησιμότητά τους.

a) **Frame**

Ένα frame είναι ένα παράθυρο ανώτατου επιπέδου με τίτλο και περίγραμμα. Το μέγεθος του frame περιλαμβάνει οποιαδήποτε περιοχή που έχει οριστεί για τα σύνορα. Οι διαστάσεις της περιοχής των συνόρων μπορούν να ληφθούν χρησιμοποιώντας τη μέθοδο getInsets, ωστόσο, εφόσον αυτές οι διαστάσεις εξαρτώνται από την πλατφόρμα, δεν είναι δυνατή η επίτευξη μιας έγκυρης τιμής εισόδου έως ότου γίνει η εμφάνιση του frame είτε από το πακέτο κλήσης είτε από την εμφάνιση.

b) **Canvas**

Μια κλάση Canvas είναι παράγωγο ή υποκατηγορία της κλάσης Component και όταν τοποθετείται πάνω σε ένα Frame, εμφανίζεται ως κενή περιοχή. Ωστόσο, για το σχεδιασμό γραφικών, μπορεί να χρησιμοποιηθεί οποιαδήποτε άλλη κλάση που προέρχεται από την κλάση Component, για παράδειγμα JPanel ή ακόμα και JTextField ή JButton.

c) **GameLoop**

Ένα GameLoop τρέχει συνεχώς κατά τη διάρκεια του παιχνιδιού. Κάθε στροφή του βρόχου, επεξεργάζεται την είσοδο του χρήστη χωρίς να εμποδίζει, ενημερώνει την κατάσταση του παιχνιδιού και καθιστά το παιχνίδι. Τα GameLoop είναι το βασικό παράδειγμα ενός "παιχνιδιού προγραμματισμού". Κάθε επανάληψη μέσω του GameLoop προχωράει την κατάσταση του παιχνιδιού κατά κάποιο ποσό. Συγκεκριμένα, εάν μετρήσουμε πόσο γρήγορα ο κύκλος παιχνιδιών κυκλώνει σε πραγματικό χρόνο, παίρνουμε τα "frames ανά δευτερόλεπτο" του παιχνιδιού. Αν το GameLoop κυλάει γρήγορα, το FPS είναι υψηλό και το παιχνίδι κινείται ομαλά και γρήγορα.

d) **Rendering**

Υπάρχουν δύο διαφορετικά είδη rendering: το ενεργό (Active) και το παθητικό (Passive). Ο κώδικας σχεδίασης τοποθετείται σε μια μέθοδο χρώματος και ο κώδικας καλείται σε απάντηση στις αιτήσεις επανακαθορισμού. Αυτά μπορεί να προέρχονται από τον ίδιο τον κώδικα, αλλά μπορεί επίσης να προέρχονται από το λειτουργικό σύστημα ως απόκριση σε γεγονότα όπως αλλαγή μεγέθους ενός παραθύρου ή κλικ σε ένα στοιχείο.

Το Active Rendering αντί να αφήσει κάποιον άλλο να αποφασίσει πότε να ζωγραφίσει, το πρόγραμμα επαναλαμβάνει συνεχώς την οθόνη σε πολύ σφιχτό while loop. Ενώ αυτό το είδος rendering δεν συνιστάται για εφαρμογές γενικού σκοπού, είναι απαραίτητο για την ανάπτυξη video games.

e) **Update**

Στον εικονικό κόσμο του παιχνιδιού διατηρείται μια συλλογή αντικειμένων. Κάθε αντικείμενο εφαρμόζει μια μέθοδο update που προσομοιώνει ένα πλαίσιο της συμπεριφοράς του αντικειμένου. Σε κάθε frame, το παιχνίδι ενημερώνει κάθε αντικείμενο της συλλογής.

f) **Mouse and Key input**

Στα παιχνίδια υπολογιστή, το πληκτρολόγιο και το ποντίκι είναι οι κύριες μέθοδοι αλληλεπίδρασης με τον υπολογιστή. Το πρόβλημα είναι ότι, ενώ η Java έχει μεγάλη υποστήριξη για αυτές τις συσκευές εισόδου για εφαρμογές GUI, τα παιχνίδια υπολογιστών πρέπει να χειρίζονται την είσοδο λίγο διαφορετικά. Παρόλο που δεν υπάρχουν ενσωματωμένες κλάσεις που να μας δίνουν αυτό που χρειαζόμαστε, μπορούμε εύκολα να δημιουργήσουμε τις δικές μας. Ακόμη, call-back functions μπορούμε να λάβουμε τη διάδραση του χρήστη μέσω του ποντικιού ή του πληκτρολογίου (Mouse and Key input
) και να τα διαχειριζόμαστε τη στιγμή που προκύπτουν.

132. Δημιουργείστε σε JAVA απλό κινούμενο αντικείμενο για το παιχνίδι σας (πχ μπαλάκι του τένις), ορίζοντας πρώτα την αρχική του θέση με x και y και κατόπιν τις επόμενες ανάλογα με την κίνηση που θέλετε να πάρει.

```
package ball_move;

import javax.swing.JFrame;
import javax.swing.WindowConstants;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.util.Scanner;
import javax.swing.JFrame;
import javax.swing.JPanel;
```

```
@SuppressWarnings("serial")

public class Ball_Move extends JPanel{

	int x = 0;
	int y = 0;

	private void moveBall(int X, int Y) {
	x = X;
	y = Y;
	}

	@Override
	public void paint(Graphics g) {
		super.paint(g);
		Graphics2D g2d = (Graphics2D) g;
		g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
			RenderingHints.VALUE_ANTIALIAS_ON);
		g2d.fillOval(x, y, 30, 30);
	}

	public static void main(String[] args) throws InterruptedException {
		int X,Y;
		Scanner reader = new Scanner(System.in);
		JFrame frame = new JFrame("Sample Frame");
		Ball_Move game = new Ball_Move();
		frame.add(game);
		frame.setSize(300, 400);
		frame.setVisible(true);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		while(true) {
			System.out.print("X:");
			X= reader.nextInt();
			System.out.println();
			System.out.print("Y:");
			Y= reader.nextInt();
			System.out.println();
			game.moveBall(X,Y);
			game.repaint();
			Thread.sleep(10);
		}
	}
}
```