


	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 1</p>

INFORME DE LABORATORIO

N° 02

INFORMACIÓN BÁSICA					
ASIGNATURA:	<i>Fundamentos de la Programación 2</i>				
TÍTULO DE LA PRÁCTICA:					
NÚMERO DE PRÁCTICA:	<i>03</i>	AÑO LECTIVO:	<i>2024-B</i>	NRO. SEMESTRE:	<i>II</i>
FECHA DE PRESENTACIÓN	<i>27/09/2024</i>	HORA DE PRESENTACIÓN	<i>20:30:05</i>		
INTEGRANTE (s) <i>Subia Huaicane Edson Fabricio</i>				NOTA (0-20)	<i>Nota colocada por el docente</i>
DOCENTE(s): <i>Lino José Pinto Oppe</i>					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <p>Actividad 1JUEGO DEL AHORCADO:</p> <p>En este ejercicio se le solicita a usted implementar el juego del ahorcado utilizando el código parcial que se le entrega. Deberá considerar que:</p> <ul style="list-style-type: none"> El juego valida el ingreso de letras solamente. En caso el usuario ingrese un carácter equivocado le dará el mensaje de error y volverá a solicitar el ingreso El juego supone que el usuario no ingresa una letra ingresada previamente El método <code>ingreseLetra()</code> debe ser modificado para incluir las consideraciones de validación Puede crear métodos adicionales. <p>En este enlace se encuentra mi repositorio y los commits que realicé para la creación y/o mejora de este programa: https://github.com/Q3son/Juego_de_Naves.git</p> <p>Mis COMMITS:</p> <ol style="list-style-type: none"> Este es el primer commit que realicé, es prácticamente la primera versión de mi código fuente que subí en GitHub.

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 2</p>

 README

Juego_de_Naves

Este es mi primer commit de Laboratorio 03, donde mostraré la primera versión de mi primer juego de Naves, un código donde no realicé mucho cambios a excepción del método para mostrar cada nave de cada flota de los equipos presentes.

Add files via upload
Beta [Give feedback](#) [Browse files](#)

 Q3son authored last week Verified

Es la primera versión de mi juego de Naves

 main
1 parent 2545871 commit caa4b45

```

+ //Método para mostrar todas las naves
+ public static void mostrarNaves(Nave [] flota){
+     for(Nave lasNaves:flota)
+         System.out.println(lasNaves);
+ }
+
+ //Método para mostrar todas las naves de un nombre que se pide por teclado
+ public static void mostrarPorNombre(Nave[] flota, Scanner sc) {
+     System.out.println("Ingrese el nombre de la nave a mostrar: ");
+     String nombreBuscado = sc.nextLine();
+
+     boolean naveEncontrada = false;
+
+     for (Nave nave : flota) {
+         if (nave.getNombre().equalsIgnoreCase(nombreBuscado)) { // Comparación insensible a mayúsculas
+             System.out.println(nave);
+             naveEncontrada = true;
+         }
+     }
+
+     if (!naveEncontrada) {
+         System.out.println("No se encontraron naves con el nombre: " + nombreBuscado);
+     }
+ }

```

Aquí destaco la primera versión de los métodos para imprimir todas las Naves y mostrarlas por Nombre a cada una de ellas.

2. Ésta es la segunda versión de mi juego (demo), donde hemos terminado de crear e implementar los métodos necesarios:

En la siguiente lista explicaré todos los cambios de esta v2.0.4:

Mejoras y ampliaciones en el sistema de batalla de naves

- Se renombra el objeto Scanner a scanPro agregar un estilo personal al código.
- Se añade el método mostrarPorNombre, el cual permite buscar y listar naves según el nombre ingresado por el usuario, mejorando la interacción con la flota.
- Se implementa el método mostrarPorPuntos, diseñado para filtrar las naves en función de un valor límite de puntos, lo que optimiza la visualización de naves según su desempeño.
- Se agrega la funcionalidad de mostrar la nave con mayor número de puntos mediante el método mostrarMayorPuntos, contribuyendo al análisis de la flota.
- Se crea el método desordenarNaves, que devuelve un nuevo arreglo con las naves reordenadas de forma aleatoria, incrementando la variabilidad en las estrategias de batalla.
- La clase Nave mantiene su estructura con métodos mutadores, accesores, y la sobrescritura del método toString() para una visualización clara de los atributos de cada nave.

```

8      - public static void main(String[] args){
9      -     Nave [] misNaves = new Nave[3];
10     -     int totalNaves = misNaves.length;
11     -     Scanner sc = new Scanner(System.in);
12     -     String nomb, col;
13     -     int fil, punt;
14     -     boolean est;
15     -
16     -     for (int i = 0; i < misNaves.length; i++) {
17     -         System.out.println("Nave " + (i+1));
18
19     +     public static void main(String[] args) {
20     +         Nave[] misNaves = new Nave[3];
21     +         int totalNaves = misNaves.length;
22     +         Scanner scanPro = new Scanner(System.in);
23     +         String nomb, col;
24     +         int fil, punt;
25     +         boolean est;
26     +
27     +         for (int i = 0; i < misNaves.length; i++) {
28     +             System.out.println("Nave " + (i + 1));

```

```



70     - //Método para mostrar todas las naves con un número de puntos inferior o igual
71     - //al número de puntos que se pide por teclado
72     - public static void mostrarPorPuntos(Nave [] flota){
73
74     +     // Método que devuelve la Nave con mayor número de Puntos
75     +     public static Nave mostrarMayorPuntos(Nave[] flota) {
76     +         Nave naveMayorPuntos = flota[0];
77
78     +         for (Nave nave : flota) {
79     +             if (nave.getPuntos() > naveMayorPuntos.getPuntos()) {
80     +                 naveMayorPuntos = nave;
81     +             }
82     +         }
83
84     +         return naveMayorPuntos;
85     +     }
86
87     - //Método que devuelve la Nave con mayor número de Puntos
88     - public static Nave mostrarMayorPuntos(Nave [] flota){
89
90     +     return naveMayorPuntos;
91     +     }

```

```


114 +     // Crear un método que devuelva un nuevo arreglo de objetos con todos los objetos
115 +     // previamente ingresados pero aleatoriamente desordenados
116 +     public static Nave[] desordenarNaves(Nave[] flota) {
117 +         Nave[] flotaDesordenada = Arrays.copyOf(flota, flota.length);
118 +         Random locoRand = new Random();
119 +
120 +         for (int i = 0; i < flotaDesordenada.length; i++) {
121 +             int randomIndex = locoRand.nextInt(flotaDesordenada.length);
122 +             Nave temp = flotaDesordenada[i];
123 +             flotaDesordenada[i] = flotaDesordenada[randomIndex];
124 +             flotaDesordenada[randomIndex] = temp;
125 +         }
126 +
127 +         return flotaDesordenada;
128 +     }

```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 4</p>

3. Para esta versión decidí mejorar mi constructor:

Para esta versión 2.0 de mi constructor-Clase Nave, realicé las siguientes mejoras

 **Q3son** committed 1 hour ago

Se ha definido un constructor con parámetros para inicializar los atributos nombre, fila, columna, estado, y puntos.



El constructor vacío (sin parámetros) fue omitido en favor del constructor completo.

Sobrescritura del método toString():

Se ha implementado el método toString() para facilitar la impresión de información de las naves.

```

8      - // Metodos mutadores
9      - public void setNombre(String n){
10     -     nombre = n;
11     - }
12     -
13     - public void setFila(int f){
14     -     fila = f;
15     - }
16     -
17     - public void setColumna(String c){
18     -     columna = c;
19     - }
20     -
21     - public void setEstado(boolean e){
22     -     estado = e;
23     - }
24     -
25     - public void setPuntos(int p){
26     -     puntos = p;
27     - }
28     -
29     - // Metodos accesores
30     -
31     - public String getNombre(){
32     -     return nombre;
33     - }
34     -
35     -
36     -
37     -
38     -
39     -
40     -
41     -
42     -
43     -
44     -
45     -
46     -
47     -
48     -
49     -
50     -
51     + // Método para mover la nave a una nueva posición
52     + public void moverNave(int nuevaFila, String nuevaColumna) {
53     +     this.fila = nuevaFila;
54     +     this.columna = nuevaColumna;
55     + }
56     +
57     + // Método para comparar si esta nave tiene más puntos que otra nave
58     + public boolean tieneMasPuntosQue(Nave otraNave) {
59     +     return this.puntos > otraNave.getPuntos();
60     + }
61     +
62     +
63     +
64     +
65     +
66     +
67     +
68     +
69     +
70     +
71     +
72     +
73     +
74     +
75     +
76     +
77     +
78     +
79     +
80     +
81     +
82     +
83     +
84     +
85     +
86     +
87     +
88     +
89     +
90     +
91     +
92     +
93     +
94     +
95     +
96     +
97     +
98     +
99     +
100    +
101    +
102    +
103    +
104    +
105    +
106    +
107    +
108    +
109    +
110    +
111    +
112    +
113    +
114    +
115    +
116    +
117    +
118    +
119    +
120    +
121    +
122    +
123    +
124    +
125    +
126    +
127    +
128    +
129    +
130    +
131    +
132    +
133    +
134    +
135    +
136    +
137    +
138    +
139    +
140    +
141    +
142    +
143    +
144    +
145    +
146    +
147    +
148    +
149    +
150    +
151    +
152    +
153    +
154    +
155    +
156    +
157    +
158    +
159    +
160    +
161    +
162    +
163    +
164    +
165    +
166    +
167    +
168    +
169    +
170    +
171    +
172    +
173    +
174    +
175    +
176    +
177    +
178    +
179    +
180    +
181    +
182    +
183    +
184    +
185    +
186    +
187    +
188    +
189    +
190    +
191    +
192    +
193    +
194    +
195    +
196    +
197    +
198    +
199    +
200    +
201    +
202    +
203    +
204    +
205    +
206    +
207    +
208    +
209    +
210    +
211    +
212    +
213    +
214    +
215    +
216    +
217    +
218    +
219    +
220    +
221    +
222    +
223    +
224    +
225    +
226    +
227    +
228    +
229    +
230    +
231    +
232    +
233    +
234    +
235    +
236    +
237    +
238    +
239    +
240    +
241    +
242    +
243    +
244    +
245    +
246    +
247    +
248    +
249    +
250    +
251    +
252    +
253    +
254    +
255    +
256    +
257    +
258    +
259    +
260    +
261    +
262    +
263    +
264    +
265    +
266    +
267    +
268    +
269    +
270    +
271    +
272    +
273    +
274    +
275    +
276    +
277    +
278    +
279    +
280    +
281    +
282    +
283    +
284    +
285    +
286    +
287    +
288    +
289    +
290    +
291    +
292    +
293    +
294    +
295    +
296    +
297    +
298    +
299    +
300    +
301    +
302    +
303    +
304    +
305    +
306    +
307    +
308    +
309    +
310    +
311    +
312    +
313    +
314    +
315    +
316    +
317    +
318    +
319    +
320    +
321    +
322    +
323    +
324    +
325    +
326    +
327    +
328    +
329    +
330    +
331    +
332    +
333    +
334    +
335    +
336    +
337    +
338    +
339    +
340    +
341    +
342    +
343    +
344    +
345    +
346    +
347    +
348    +
349    +
350    +
351    +
352    +
353    +
354    +
355    +
356    +
357    +
358    +
359    +
360    +
361    +
362    +
363    +
364    +
365    +
366    +
367    +
368    +
369    +
370    +
371    +
372    +
373    +
374    +
375    +
376    +
377    +
378    +
379    +
380    +
381    +
382    +
383    +
384    +
385    +
386    +
387    +
388    +
389    +
390    +
391    +
392    +
393    +
394    +
395    +
396    +
397    +
398    +
399    +
400    +
401    +
402    +
403    +
404    +
405    +
406    +
407    +
408    +
409    +
410    +
411    +
412    +
413    +
414    +
415    +
416    +
417    +
418    +
419    +
420    +
421    +
422    +
423    +
424    +
425    +
426    +
427    +
428    +
429    +
430    +
431    +
432    +
433    +
434    +
435    +
436    +
437    +
438    +
439    +
440    +
441    +
442    +
443    +
444    +
445    +
446    +
447    +
448    +
449    +
450    +
451    +
452    +
453    +
454    +
455    +
456    +
457    +
458    +
459    +
460    +
461    +
462    +
463    +
464    +
465    +
466    +
467    +
468    +
469    +
470    +
471    +
472    +
473    +
474    +
475    +
476    +
477    +
478    +
479    +
480    +
481    +
482    +
483    +
484    +
485    +
486    +
487    +
488    +
489    +
490    +
491    +
492    +
493    +
494    +
495    +
496    +
497    +
498    +
499    +
500    +
501    +
502    +
503    +
504    +
505    +
506    +
507    +
508    +
509    +
510    +
511    +
512    +
513    +
514    +
515    +
516    +
517    +
518    +
519    +
520    +
521    +
522    +
523    +
524    +
525    +
526    +
527    +
528    +
529    +
530    +
531    +
532    +
533    +
534    +
535    +
536    +
537    +
538    +
539    +
540    +
541    +
542    +
543    +
544    +
545    +
546    +
547    +
548    +
549    +
550    +
551    +
552    +
553    +
554    +
555    +
556    +
557    +
558    +
559    +
560    +
561    +
562    +
563    +
564    +
565    +
566    +
567    +
568    +
569    +
570    +
571    +
572    +
573    +
574    +
575    +
576    +
577    +
578    +
579    +
580    +
581    +
582    +
583    +
584    +
585    +
586    +
587    +
588    +
589    +
590    +
591    +
592    +
593    +
594    +
595    +
596    +
597    +
598    +
599    +
600    +
601    +
602    +
603    +
604    +
605    +
606    +
607    +
608    +
609    +
610    +
611    +
612    +
613    +
614    +
615    +
616    +
617    +
618    +
619    +
620    +
621    +
622    +
623    +
624    +
625    +
626    +
627    +
628    +
629    +
630    +
631    +
632    +
633    +
634    +
635    +
636    +
637    +
638    +
639    +
640    +
641    +
642    +
643    +
644    +
645    +
646    +
647    +
648    +
649    +
650    +
651    +
652    +
653    +
654    +
655    +
656    +
657    +
658    +
659    +
660    +
661    +
662    +
663    +
664    +
665    +
666    +
667    +
668    +
669    +
670    +
671    +
672    +
673    +
674    +
675    +
676    +
677    +
678    +
679    +
680    +
681    +
682    +
683    +
684    +
685    +
686    +
687    +
688    +
689    +
690    +
691    +
692    +
693    +
694    +
695    +
696    +
697    +
698    +
699    +
700    +
701    +
702    +
703    +
704    +
705    +
706    +
707    +
708    +
709    +
710    +
711    +
712    +
713    +
714    +
715    +
716    +
717    +
718    +
719    +
720    +
721    +
722    +
723    +
724    +
725    +
726    +
727    +
728    +
729    +
730    +
731    +
732    +
733    +
734    +
735    +
736    +
737    +
738    +
739    +
740    +
741    +
742    +
743    +
744    +
745    +
746    +
747    +
748    +
749    +
750    +
751    +
752    +
753    +
754    +
755    +
756    +
757    +
758    +
759    +
760    +
761    +
762    +
763    +
764    +
765    +
766    +
767    +
768    +
769    +
770    +
771    +
772    +
773    +
774    +
775    +
776    +
777    +
778    +
779    +
780    +
781    +
782    +
783    +
784    +
785    +
786    +
787    +
788    +
789    +
790    +
791    +
792    +
793    +
794    +
795    +
796    +
797    +
798    +
799    +
800    +
801    +
802    +
803    +
804    +
805    +
806    +
807    +
808    +
809    +
810    +
811    +
812    +
813    +
814    +
815    +
816    +
817    +
818    +
819    +
820    +
821    +
822    +
823    +
824    +
825    +
826    +
827    +
828    +
829    +
830    +
831    +
832    +
833    +
834    +
835    +
836    +
837    +
838    +
839    +
840    +
841    +
842    +
843    +
844    +
845    +
846    +
847    +
848    +
849    +
850    +
851    +
852    +
853    +
854    +
855    +
856    +
857    +
858    +
859    +
860    +
861    +
862    +
863    +
864    +
865    +
866    +
867    +
868    +
869    +
870    +
871    +
872    +
873    +
874    +
875    +
876    +
877    +
878    +
879    +
880    +
881    +
882    +
883    +
884    +
885    +
886    +
887    +
888    +
889    +
890    +
891    +
892    +
893    +
894    +
895    +
896    +
897    +
898    +
899    +
900    +
901    +
902    +
903    +
904    +
905    +
906    +
907    +
908    +
909    +
910    +
911    +
912    +
913    +
914    +
915    +
916    +
917    +
918    +
919    +
920    +
921    +
922    +
923    +
924    +
925    +
926    +
927    +
928    +
929    +
930    +
931    +
932    +
933    +
934    +
935    +
936    +
937    +
938    +
939    +
940    +
941    +
942    +
943    +
944    +
945    +
946    +
947    +
948    +
949    +
950    +
951    +
952    +
953    +
954    +
955    +
956    +
957    +
958    +
959    +
960    +
961    +
962    +
963    +
964    +
965    +
966    +
967    +
968    +
969    +
970    +
971    +
972    +
973    +
974    +
975    +
976    +
977    +
978    +
979    +
980    +
981    +
982    +
983    +
984    +
985    +
986    +
987    +
988    +
989    +
990    +
991    +
992    +
993    +
994    +
995    +
996    +
997    +
998    +
999    +
1000    +
1001    +
1002    +
1003    +
1004    +
1005    +
1006    +
1007    +
1008    +
1009    +
1010    +
1011    +
1012    +
1013    +
1014    +
1015    +
1016    +
1017    +
1018    +
1019    +
1020    +
1021    +
1022    +
1023    +
1024    +
1025    +
1026    +
1027    +
1028    +
1029    +
1030    +
1031    +
1032    +
1033    +
1034    +
1035    +
1036    +
1037    +
1038    +
1039    +
1040    +
1041    +
1042    +
1043    +
1044    +
1045    +
1046    +
1047    +
1048    +
1049    +
1050    +
1051    +
1052    +
1053    +
1054    +
1055    +
1056    +
1057    +
1058    +
1059    +
1060    +
1061    +
1062    +
1063    +
1064    +
1065    +
1066    +
1067    +
1068    +
1069    +
1070    +
1071    +
1072    +
1073    +
1074    +
1075    +
1076    +
1077    +
1078    +
1079    +
1080    +
1081    +
1082    +
1083    +
1084    +
1085    +
1086    +
1087    +
1088    +
1089    +
1090    +
1091    +
1092    +
1093    +
1094    +
1095    +
1096    +
1097    +
1098    +
1099    +
1100    +
1101    +
1102    +
1103    +
1104    +
1105    +
1106    +
1107    +
1108    +
1109    +
1110    +
1111    +
1112    +
1113    +
1114    +
1115    +
1116    +
1117    +
1118    +
1119    +
1120    +
1121    +
1122    +
1123    +
1124    +
1125    +
1126    +
1127    +
1128    +
1129    +
1130    +
1131    +
1132    +
1133    +
1134    +
1135    +
1136    +
1137    +
1138    +
1139    +
1140    +
1141    +
1142    +
1143    +
1144    +
1145    +
1146    +
1147    +
1148    +
1149    +
1150    +
1151    +
1152    +
1153    +
1154    +
1155    +
1156    +
1157    +
1158    +
1159    +
1160    +
1161    +
1162    +
1163    +
1164    +
1165    +
1166    +
1167    +
1168    +
1169    +
1170    +
1171    +
1172    +
1173    +
1174    +
1175    +
1176    +
1177    +
1178    +
1179    +
1180    +
1181    +
1182    +
1183    +
1184    +
1185    +
1186    +
1187    +
1188    +
1189    +
1190    +
1191    +
1192    +
1193    +
1194    +
1195    +
1196    +
1197    +
1198    +
1199    +
1200    +
1201    +
1202    +
1203    +
1204    +
1205    +
1206    +
1207    +
1208    +
1209    +
1210    +
1211    +
1212    +
1213    +
1214    +
1215    +
1216    +
1217    +
1218    +
1219    +
1220    +
1221    +
1222    +
1223    +
1224    +
1225    +
1226    +
1227    +
1228    +
1229    +
1230    +
1231    +
1232    +
1233    +
1234    +
1235    +
1236    +
1237    +
1238    +
1239    +
1240    +
1241    +
1242    +
1243    +
1244    +
1245    +
1246    +
1247    +
1248    +
1249    +
1250    +
1251    +
1252    +
1253    +
1254    +
1255    +
1256    +
1257    +
1258    +
1259    +
1260    +
1261    +
1262    +
1263    +
1264    +
1265    +
1266    +
1267    +
1268    +
1269    +
1270    +
1271    +
1272    +
1273    +
1274    +
1275    +
1276    +
1277    +
1278    +
1279    +
1280    +
1281    +
1282    +
1283    +
1284    +
1285    +
1286    +
1287    +
1288    +
1289    +
1290    +
1291    +
1292    +
1293    +
1294    +
1295    +
1296    +
1297    +
1298    +
1299    +
1300    +
1301    +
1302    +
1303    +
1304    +
1305    +
1306    +
1307    +
1308    +
1309    +
1310    +
1311    +
1312    +
1313    +
1314    +
1315    +
1316    +
1317    +
1318    +
1319    +
1320    +
1321    +
1322    +
1323    +
1324    +
1325    +
1326    +
1327    +
1328    +
1329    +
1330    +
1331    +
1332    +
1333    +
1334    +
1335    +
1336    +
1337    +
1338    +
1339    +
1340    +
1341    +
1342    +
1343    +
1344    +
1345    +
1346    +
1347    +
1348    +
1349    +
1350    +
1351    +
1352    +
1353    +
1354    +
1355    +
1356    +
1357    +
1358    +
1359    +
1360    +
1361    +
1362    +
1363    +
1364    +
1365    +
1366    +
1367    +
1368    +
1369    +
1370    +
1371    +
1372    +
1373    +
1374    +
1375    +
1376    +
1377    +
1378    +
1379    +
1380    +
1381    +
1382    +
1383    +
1384    +
1385    +
1386    +
1387    +
1388    +
1389    +
1390    +
1391    +
1392    +
1393    +
1394    +
1395    +
1396    +
1397    +
1398    +
1399    +
1400    +
1401    +
1402    +
1403    +
1404    +
1405    +
1406    +
1407    +
1408    +
1409    +
1410    +
1411    +
1412    +
1413    +
1414    +
1415    +
1416    +
1417    +
1418    +
1419    +
1420    +
1421    +
1422    +
1423    +
1424    +
1425    +
1426    +
1427    +
1428    +
1429    +
1430    +
1431    +
1432    +
1433    +
1434    +
1435    +
1436    +
1437    +
1438    +
1439    +
1440    +
1441    +
1442    +
1443    +
1444    +
1445    +
1446    +
1447    +
1448    +
1449    +
1450    +
1451    +
1452    +
1453    +
1454    +
1455    +
1456    +
1457    +
1458    +
1459    +
1460    +
1461    +
1462    +
1463    +
1464    +
1465    +
1466    +
1467    +
1468    +
1469    +
1470    +
1471    +
1472    +
1473    +
1474    +
1475    +
1476    +
1477    +
1478    +
1479    +
1480    +
1481    +
1482    +
1483    +
1484    +
1485    +
1486    +
1487    +
1488    +
1489    +
1490    +
1491    +
1492    +
1493    +
1494    +
1495    +
1496    +
1497    +
1498    +
1499    +
1500    +
1501    +
1502    +
1503    +
1504    +
1505    +
1506    +
1507    +
1508    +
1509    +
1510    +
1511    +
1512    +
1513    +
1514    +
1515    +
1516    +
1517    +
1518    +
1519    +
1520    +
1521    +
1522    +
1523    +
1524    +
1525    +
1526    +
1527    +
1528    +
1529    +
1530    +
1531    +
1532    +
1533    +
1534    +
1535    +
1536    +
1537    +
1538    +
1539    +
1540    +
1541    +
1542    +
1543    +
1544    +
1545    +
1546    +
1547    +
1548    +
1549    +
1550    +
1551    +
1552    +
1553    +
1554    +
1555    +
1556    +
1557    +
1558    +
1559    +
1560    +
1561    +
1562    +
1563    +
1564    +
1565    +
1566    +
1567    +
1568    +
1569    +
1570    +
1571    +
1572    +
1573    +
1574    +
1575    +
1576    +
1577    +
1578    +
1579    +
1580    +
1581    +
1582    +
1583    +
1584    +
1585    +
1586    +
1587    +
1588    +
1589    +
1590    +
1591    +
1592    +
1593    +
1594    +
1595    +
1596    +
1597    +
1598    +
1599    +
1600    +
1601    +
1602    +
1603    +
1604    +
1605    +
1606    +
1607    +
1608    +
1609    +
1610    +
1611    +
1612    +
1613    +
1614    +
1615    +
1616    +
1617    +
1618    +
1619    +
1620    +
1621    +
1622    +
1623    +
1624    +
1625    +
1626    +
1627    +
1628    +
1629    +
1630    +
1631    +
1632    +
1633    +
1634    +
1635    +
1636    +
1637    +
1638    +
1639    +
1640    +
1641    +
1642    +
1643    +
1644    +
1645    +
1646    +
1647    +
1648    +
1649    +
1650    +
1651    +
1652    +
1653    +
1654    +
1655    +
1656    +
1657    +
1658    +
1659    +
1660    +
1661    +
1662    +
1663    +
1664    +
1665    +
1666    +
1667    +
1668    +
1669    +
1670    +
1671    +
1672    +
1673    +
1674    +
1675    +
1676    +
1677    +
1678    +
1679    +
1680    +
1681    +
1682    +
1683    +
1684    +
1685    +
1686    +
1687    +
1688    +
1689    +
1690    +
1691    +
1692    +
1693    +
1694    +
1695    +
1696    +
1697    +
1698    +
1699    +
1700    +
1701    +
1702    +
1703    +
1704    +
1705    +
1706    +
1707    +
1708    +
1709    +
1710    +
1711    +
1712    +
1713    +
1714    +
1715    +
1716    +
1717    +
1718    +
1719    +
1720    +
1721    +
1722    +
1723    +
1724    +
1725    +
1726    +
1727    +
1728    +
1729    +
1730    +
1731    +
1732    +
1733    +
1734    +
1735    +
1736    +
1737    +
1738    +
1739    +
1740    +
1741    +
1742    +
1743    +
1744    +
1745    +
1746    +
1747    +
1748    +
1749    +
1750    +
1751    +
1752    +
1753    +
1754    +
1755    +
1756    +
1757    +
1758    +
1759    +
1760    +
1761    +
1762    +
1763    +
1764    +
1765    +
1766    +
1767    +
1768    +
1769    +
1770    +
1771    +
1772    +
1773    +
1774    +
1775    +
1776    +
1777    +
1778    +
1
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 5</p>

```

107 - // Crear un método que devuelva un nuevo arreglo de objetos con todos los objetos
108 - // previamente ingresados pero aleatoriamente desordenados
109 + // 12. Método que desordena aleatoriamente un arreglo de naves
109 110 public static Nave[] desordenarNaves(Nave[] flota) {
110 111     Nave[] flotaDesordenada = Arrays.copyOf(flota, flota.length);
111 112     Random locoRand = new Random();
112 113
113 114     @@ -119,4 +120,4 @@ public static Nave[] desordenarNaves(Nave[] flota) {
119 120
120 121     return flotaDesordenada;
121 122 }

```

```

33 + // 3. Mostrar las naves creadas

```

```

+ // 7. Mostrar naves desordenadas aleatoriamente

```

```

96 + // 11. Método que devuelve la Nave con mayor número de Puntos



```

Aquí decidí enumerar los comentarios más importantes para una mejor explicación de los métodos presente en el Programa DemoBatalla, Juego de naves.

```

DemoBatalla.java
@@ -5,13 +5,14 @@
5 5 */
6 6 public class DemoBatalla {
7 7     public static void main(String[] args) {
8 - Nave[] misNaves = new Nave[3]; // Crear un arreglo de naves
9 - Scanner scanPro = new Scanner(System.in); // Lector de entradas
8 + // 1. Crear un arreglo de naves
9 + Nave[] misNaves = new Nave[3];
10 + Scanner scanPro = new Scanner(System.in);
10 11 String nomb, col;
11 12 int fil, punt;
12 13 boolean est;
13 14
14 - // Recolección de datos y creación de cada nave
15 + // 2. Recolección de datos y creación de cada nave
15 16 for (int i = 0; i < misNaves.length; i++) {
16 17     System.out.println("Nave " + (i + 1));
17 18     System.out.print("Nombre: ");
18 19
19 20     @@ -29,30 +30,32 @@ public static void main(String[] args) {
29 30     misNaves[i] = new Nave(nomb, fil, col, est, punt);
30 31 }
31 32

```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 6</p>

En la siguiente sección mostraré la versión final de mi código fuente del programa, trabajado en Visual Studio, en cada captura de pantalla se visualizarán líneas de código fundamentadas correctamente y un programa bien elaborado, y la respectiva ejecución del programa final. – Se ocuparon 111 líneas de código en total.

```



You, 48 minutes ago | 1 author (You)
1  import java.util.*;
You, 48 minutes ago | 1 author (You)
2  /* Autor: Subia Huacane Edson Fabricio
3   * Colaborador: Mi persona
4   * Tiempo: 40 minutos
5   */
6  public class DemoBatalla {
    Run | Debug
7      public static void main(String[] args) {
8          // 1. Crear un arreglo de naves
9          Nave[] misNaves = new Nave[3];
10         Scanner scanPro = new Scanner(System.in);
11         String nomb, col;
12         int fil, punt;
13         boolean est;
14
15         // 2. Recolección de datos y creación de cada nave
16         for (int i = 0; i < misNaves.length; i++) {
17             System.out.println("Nave " + (i + 1));
18             System.out.print(s:"Nombre: ");
19             nomb = scanPro.next(); // Capturar nombre
20             System.out.print(s:"Fila: ");
21             fil = scanPro.nextInt(); // Capturar fila
22             System.out.print(s:"Columna: ");
23             col = scanPro.next(); // Capturar columna
24             System.out.print(s:"Estado (true para operativa, false para inactiva): ");
25             est = scanPro.nextBoolean(); // Capturar estado
26             System.out.print(s:"Puntos: ");
27             punt = scanPro.nextInt(); // Capturar puntos
28
29             // Crear cada nave usando el constructor con parámetros
30             misNaves[i] = new Nave(nomb, fil, col, est, punt);
31         }
32
33         // 3. Mostrar las naves creadas
34         System.out.println(x:"\nNaves creadas:");
35         mostrarNaves(misNaves);
36
37         // 4. Solicitar y mostrar naves por nombre
38         mostrarPorNombre(misNaves, scanPro);
39
40         // 5. Mostrar naves según los puntos ingresados
41         mostrarPorPuntos(misNaves, scanPro);
42
43         // 6. Mostrar nave con mayor número de puntos
44         System.out.println("\nNave con mayor número de puntos: " + mostrarMayorPuntos(misNaves));
45
46         // 7. Mostrar naves desordenadas aleatoriamente
47         Nave[] navesDesordenadas = desordenarNaves(misNaves);
48         System.out.println(x:"\nNaves desordenadas aleatoriamente:");
49         mostrarNaves(navesDesordenadas);
50     }
51
52     // 8. Método para mostrar todas las naves
53     public static void mostrarNaves(Nave[] flota) {
54         for (Nave lasNaves : flota)
55             System.out.println(lasNaves);
56     }
57
58     // 9. Método para mostrar todas las naves de un nombre que se pide por teclado
59     public static void mostrarPorNombre(Nave[] flota, Scanner scanPro) {
60         System.out.println(x:"Ingrese el nombre de la nave a mostrar: ");
61         String nombreBuscado = scanPro.next(); // Cambiar a next() para evitar problemas con espacios
62
63         boolean naveEncontrada = false;
64
65         for (Nave nave : flota) {
66             if (nave.getNombre().equalsIgnoreCase(nombreBuscado)) { // Comparación insensible a mayúsculas
67                 System.out.println(nave);
68                 naveEncontrada = true;

```

```
69     }
70 }
71
72 if (!naveEncontrada) {
73     System.out.println("No se encontraron naves con el nombre: " + nombreBuscado);
74 }
75 }
76
77 // 10. Método para mostrar naves con puntos inferiores o iguales a los ingresados
78 public static void mostrarPorPuntos(Nave[] flota, Scanner scanPro) {
79     System.out.print(s:"Ingrese el límite de puntos: ");
80     int puntosMax = scanPro.nextInt();
81
82     boolean naveEncontrada = false;
83
84     for (Nave nave : flota) {
85         if (nave.getPuntos() <= puntosMax) {
86             System.out.println(nave);
87             naveEncontrada = true;
88         }
89     }
90
91     if (!naveEncontrada) {
92         System.out.println("No se encontraron naves con puntos menores o iguales a: " + puntosMax);
93     }
94 }
95
96 // 11. Método que devuelve la Nave con mayor número de Puntos
97 public static Nave mostrarMayorPuntos(Nave[] flota) {
98     Nave naveMayorPuntos = flota[0];
99
100    for (Nave nave : flota) {
101        if (nave.getPuntos() > naveMayorPuntos.getPuntos()) {
102            naveMayorPuntos = nave;
103        }
104    }
```

```
104    }
105
106    return naveMayorPuntos;
107 }
108
109 // 12. Método que desordena aleatoriamente un arreglo de naves
110 public static Nave[] desordenarNaves(Nave[] flota) {
111     Nave[] flotaDesordenada = Arrays.copyOf(flota, flota.length);
112     Random locoRand = new Random();
113
114     for (int i = 0; i < flotaDesordenada.length; i++) {
115         int randomIndex = locoRand.nextInt(flotaDesordenada.length);
116         Nave temp = flotaDesordenada[i];
117         flotaDesordenada[i] = flotaDesordenada[randomIndex];
118         flotaDesordenada[randomIndex] = temp;
119     }
120
121     return flotaDesordenada;
122 }
123 }
```

You, 48 minutes ago • Para esta versión final (v3.5.0) realicé los c...

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>

EJECUCIÓN DEL PROGRAMA (v3.0.0): "JUEGO DEL AHORCADO"

```
er\workspaceStorage\bab0a6105cf282749fa0a4c3a1b0d37\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'DemoBatalla'
Nave 1
Nombre: XD
Fila: 3
Columna: 4
Estado (true para operativa, false para inactiva): True
Puntos: 10
Nave 2
Nombre: Xd
Fila: 4
Columna: 4
Estado (true para operativa, false para inactiva): True
Puntos: 9
Nave 3
Nombre: ROLLO
Fila: 5
Columna: 6
Estado (true para operativa, false para inactiva): False
Puntos: 11

Naves creadas:
Nave [Nombre: XD, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]
Nave [Nombre: Xd, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]
Nave [Nombre: ROLLO, Fila: 5, Columna: 6, Estado: Inactiva, Puntos: 11]
Ingrese el nombre de la nave a mostrar:
XD
Nave [Nombre: XD, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]
Nave [Nombre: Xd, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]
Ingrese el límite de puntos: 9
Nave [Nombre: Xd, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]

Nave con mayor número de puntos: Nave [Nombre: ROLLO, Fila: 5, Columna: 6, Estado: Inactiva, Puntos: 11]

Naves desordenadas aleatoriamente:
Nave [Nombre: ROLLO, Fila: 5, Columna: 6, Estado: Inactiva, Puntos: 11]
Nave [Nombre: Xd, Fila: 4, Columna: 4, Estado: Operativa, Puntos: 9]
Nave [Nombre: XD, Fila: 3, Columna: 4, Estado: Operativa, Puntos: 10]
PS C:\Users\Edson\Desktop\2do semestres 2024 Edson>
```

5. PRUEBAS



¿Con qué valores comprobaste que tu práctica estuviera correcta?

Comprobé la práctica utilizando diversas combinaciones de entrada al crear las naves en el juego. Por ejemplo, ingresé nombres de naves como "XD", "Xd" y "Rollo", y varié los valores para la fila, columna, estado y puntos. También probé con diferentes estados (true y false) para verificar la funcionalidad de cambio de estado. Además, al ejecutar las búsquedas de naves, utilicé nombres existentes y no existentes, así como diferentes límites de puntos.

¿Qué resultado esperabas obtener para cada valor de entrada?

Esperaba que el juego:

- Al crear naves: se inicializaran correctamente con los valores proporcionados, y que la información de cada nave se mostrara de forma precisa.*

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 9</p>

- *Al buscar naves por nombre: el programa encontrara correctamente las naves que coincidieran con el nombre ingresado, y si no existían, mostrara un mensaje informativo.*
- *Al buscar naves por puntos: se mostraran solo aquellas naves que cumplieran con el límite de puntos especificado.*
- *Al mostrar la nave con mayor cantidad de puntos: se identificara correctamente la nave que tuviera el mayor puntaje en el arreglo.*
- *Al desordenar naves: se generará un nuevo arreglo con las naves en un orden aleatorio.*

¿Qué valor o comportamiento obtuviste para cada valor de entrada?

Los resultados obtenidos fueron los esperados:



- *Al crear naves: todas las naves se inicializaron correctamente y la información se mostró con el formato adecuado.*
- *Al buscar naves por nombre: el programa encontró las naves correctamente cuando se ingresaron nombres válidos y mostró un mensaje cuando no se encontró coincidencia.*
- *Al buscar naves por puntos: se mostraron únicamente las naves que cumplían con el límite de puntos ingresado, o se informó si ninguna cumplía con la condición.*
- *Al mostrar la nave con mayor cantidad de puntos: el programa identificó correctamente la nave con el mayor puntaje en el arreglo.*
- *Al desordenar naves: se creó un nuevo arreglo con las naves en un orden aleatorio, validando así la funcionalidad de este método.*

6. CUESTIONARIO:

¿Cuáles fueron los datos más importantes que consideró para realizar el código?

Los datos más importantes que consideré para realizar el código del juego de Naves fueron los siguientes:

1. **Nave:** Cada nave tiene atributos como nombre, fila, columna, estado y puntos. Estos datos son esenciales para definir las características y el estado de cada nave en el juego.
2. **Arreglo de naves:** Se utiliza un arreglo para almacenar múltiples objetos de tipo Nave. Esto permite gestionar y manipular varias naves de manera eficiente durante la partida.
3. **Estado de la nave:** El atributo estado indica si la nave está operativa o inactiva. Este dato es fundamental para el funcionamiento del juego, ya que afecta las decisiones del jugador y el resultado del juego.
4. **Interacción con el usuario:** Se recopilan datos como el nombre de la nave, fila, columna, estado y puntos mediante un Scanner. Esta entrada es crucial para personalizar la experiencia del juego y permitir al jugador interactuar con las naves.
5. **Métodos de comparación y actualización:** Métodos como `tieneMasPuntosQue()` y `actualizarPuntos()` son importantes para implementar la lógica del juego y gestionar el puntaje de las naves.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 10</p>

CONCLUSIONES

Colocar las conclusiones, apreciaciones reflexivas, opiniones finales a cerca de los resultados obtenidos de la sesión de laboratorio.

El desarrollo del juego de Naves evidencia una estructura clara y una interacción fluida que enriquecen la experiencia del usuario. La implementación de métodos para gestionar las naves y sus atributos permitió un manejo efectivo de la lógica del juego, garantizando que las funcionalidades respondieran de manera adecuada a las entradas del usuario. Los resultados obtenidos coincidieron con las expectativas, lo que confirma la eficacia de la programación realizada. Este proyecto no solo facilitó la aplicación de conceptos fundamentales de programación orientada a objetos, sino que también propició un aprendizaje significativo sobre el diseño de juegos y la importancia de la interacción en la experiencia del usuario.

METODOLOGÍA DE TRABAJO

Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el procedimiento/secuencia de pasos en forma general.

- a) **Comprensión del problema:** En esta etapa, revisé cada una de las actividades propuestas, identificando cuidadosamente las restricciones y los objetivos a alcanzar.
- b) **Diseño del algoritmo:** Planifiqué la secuencia lógica necesaria para implementar la solución, aplicando los conocimientos adquiridos en Fundamentos de Programación I y II.
- c) **Codificación:** Procedí a implementar los programas solicitados, asegurándome de utilizar correctamente los arreglos y métodos.
- d) **Pruebas:** Realicé pruebas adicionales para verificar que el código funcionara de manera correcta con diferentes casos de prueba.

REFERENCIAS Y BIBLIOGRAFÍA

Colocare las referencias utilizadas para el desarrollo de la práctica en formato IEEE

M. W. Aedo López, *Fundamentos de programación I: Java Básico*, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, Jul. 2019. ISBN: 978-612-4337-55-0. 116 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 21 x 29.7 cm

[https://github.com/LINOPINTO2023/FundProg2/blob/main/entregaLaboratorio01/Hilacondo Emanuel LABORATORIO_01.pdf](https://github.com/LINOPINTO2023/FundProg2/blob/main/entregaLaboratorio01/Hilacondo_Emanuel_LABORATORIO_01.pdf)

https://github.com/Q3son/Juego_de_Naves.git

RÚBRICA PARA EL CONTENIDO DEL INFORME Y DEMOSTRACIÓN

El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumplió con el ítem correspondiente.

Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.

El alumno debe autocalificarse en la columna Estudiante de acuerdo con la siguiente tabla:

Tabla 1: Niveles de desempeño

Nivel				
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	0	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
TOTAL		20	8	17	