Nome: Número:

1º Teste de "Introdução à Programação" (2014/15 – 1º Semestre)

Duração 2H

Instruções importantes:

- Responda a cada pergunta no espaço atribuído para o efeito.
- Verifique que **todas** as folhas estão identificadas com o seu nome e número.
- Antes de começar a resolver, leia o enunciado do **princípio até ao fim**.
- **Pode** usar caneta ou lápis.
- Não é permitido consultar quaisquer elementos para além deste enunciado.
- Qualquer tentativa de fraude comprovada acarretará a reprovação na disciplina.
- Não é permitido sair da sala antes que o teste termine.

Nome: Número:

Grupo I

Para cada um dos excertos de programas Java apresentados, assinale **que linhas são executadas**, e indique qual o valor que **cada uma** das variáveis contém após **cada linha** ser executada (sugestão: para cada alínea, faça uma tabela cujas linhas são as linhas do programa e cujas colunas são as variáveis do programa; assinale as linhas não executadas com "**NE**"). **(4 valores)**

a) Execução de f(10,3)

```
1   boolean f(int x, int y) {
2     boolean pred = y < x % y;
3     return pred;
4     x = y * x;
5     pred = pred || x > y;
6   }
```

b)

```
1   {
2     int t = 9000;
3     int a = t / 3600;
4     int b = t % 3600;
5     int c = b / 60;
6     int  d = b % 60;
7     System.out.println( b + ":" + d);
8   }
```

Nome: Número:

c) Execução de doSomething(70) com this.a == 59, this.b == 13, this.c == 8, this.d == 123

```
1  void doSomething(int arg) {
2     if(this.a < arg) {
3        this.a = arg;
4        this.c++;
5     }
6     else {
7        this.b = arg;
8        this.c++;
9     }
10    this.d += arg;
11 }
```

d)

Existe no excerto de programa apresentado em c) uma repetição de código. Indique-a, usando para isso os números de linha e sugira uma versão do código sem a repetição e que efectua as mesmas acções.

Nome: Número:

Grupo II

Pretende-se a construção de uma classe Game para criar objectos que oferecem um jogo de tabuleiro para 2 participantes.

Os jogadores são identificados pelos números 1 e 2, respectivamente. O tabuleiro é composto por uma sequência de casas, numeradas da 1 à 120, onde existe uma casa denotada por casa da morte. A casa da morte é indicada no momento de criação do jogo.

Para jogar, um participante lança um dado, e o peão desse jogador avança o número de casas indicado no dado. Note que, em caso de ultrapassar a última casa (120) numa jogada, o peão deve voltar ao tabuleiro. Por exemplo, se o jogador estava na casa 119 e saiu 3 no dado, o peão avança para a 120 e recua 2 casas, ficando na 118.

O jogo termina quando um dos participantes chega à última casa ou à casa da morte. O vencedor do jogo é o participante que chegou à última casa ou o adversário do que chegou à casa da morte.

Note que neste jogo é sempre possível saber o número de vezes que um dado participante jogou, já que os participantes não jogam intercaladamente, como acontece na maioria dos jogos de tabuleiro.

A. Complete a seguinte implementação da classe Game **(7 valores)**:

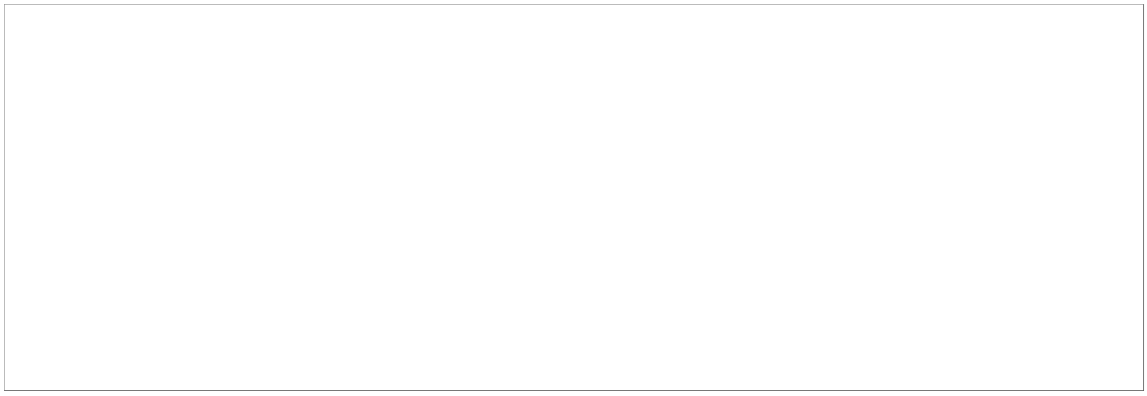
```
import java.util.Random;

public class Game {

	/* Constantes */

	public static final int MAX= 6;
```

```
	/* Variáveis de Instância */
```

Nome: Número:

```
/* Construtor
 * @param deathPos: posicao na pista do jogo que mata o jogador
 * @pre deathPos >=1 && deathPos < MAXPOS
 */
public Game(int deathPos){
```

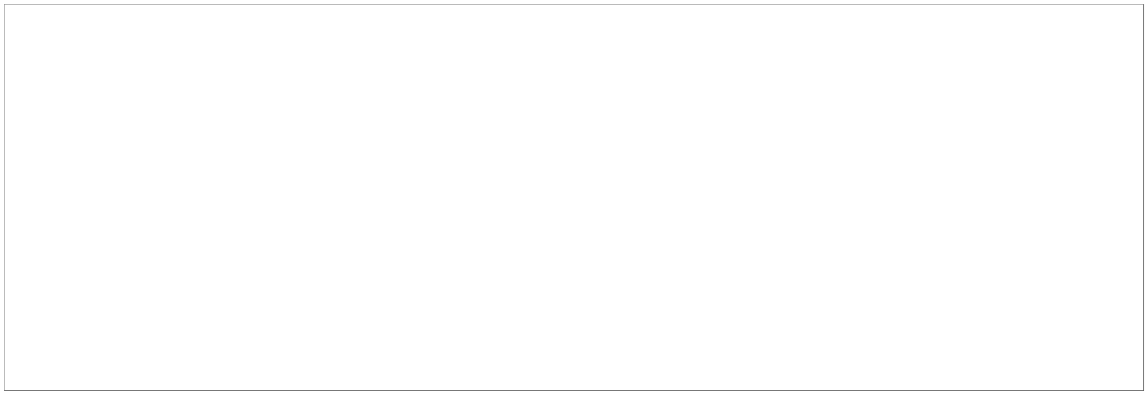
```
}
/*
 * Indica se o jogo terminou: Um dos jogadores morreu ou chegou à posição MAXPOS
 * @return true se jogo acabou; false caso contrário
 */
public boolean gameEnd(){
```



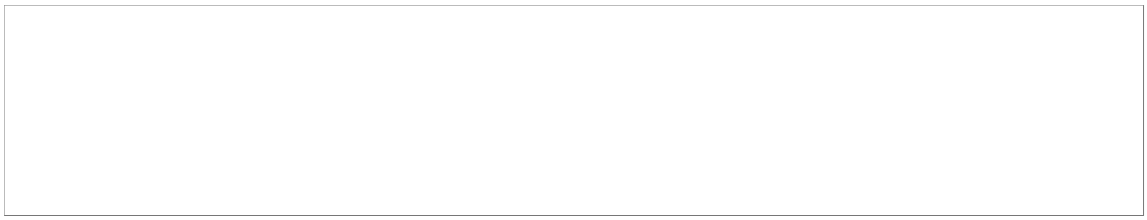
```
}
```

Nome: Número:

```
/*
 * Consulta do vencedor
 * @pre gameEnd()
 * @return 1, caso jogador 1 e 2 caso jogador 2
 */
public int winner(){
```



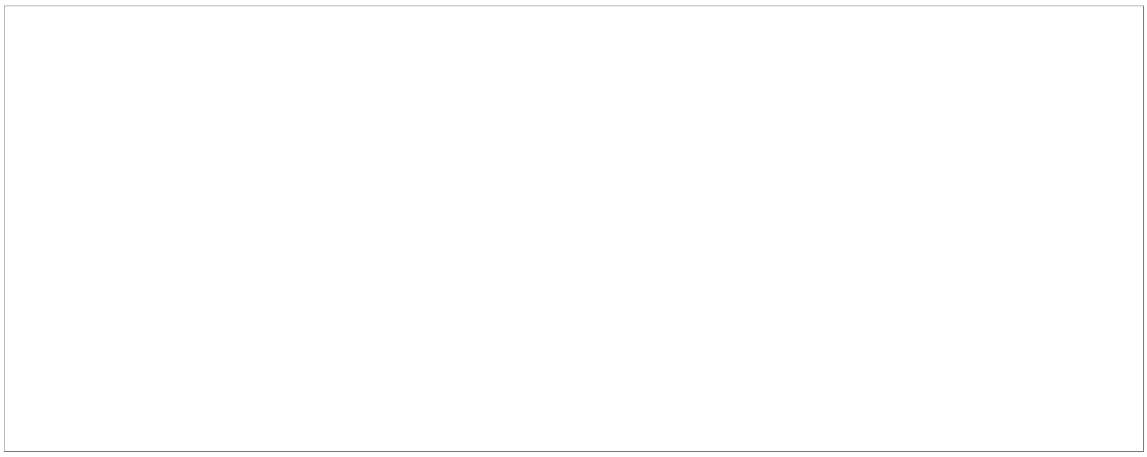
```
}
/*
 * Consulta o número de jogadas do jogador
 * @param player: jogador
 * @pre player ==1 ou player ==2
 * @return numero de jogadas
 */
public int moves(int player){
```



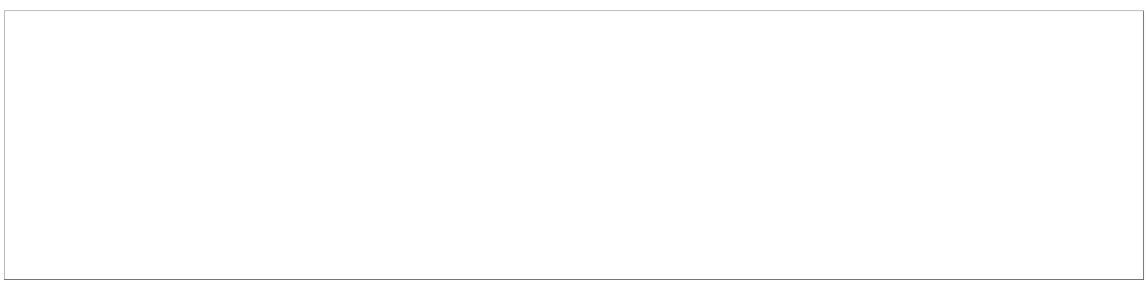
```
}
```

Nome: Número:

```
        /*
         * Jogar: lançar o dado e avançar as casas
         * @param player: jogador
         * @pre player ==1 ou player ==2
         * @return posicao do jogador ou zero, caso o jogador fique na posição da morte
         */
        public int play(int player){
              int die = Game.genNumber(); // Simula lançamento de dado
```



```
        }
        /* Métodos privados */
        /* Gera um número entre 1 e 6
         */
        private static int genNumber() {
            Random r = new Random();
            return r.nextInt(MAX) + 1;
          }
```



```
}
```

Nome: Número:

B. Complete o seguinte programa principal, que simula um jogo. O jogo é jogado pelos jogadores até que acaba, sendo o utilizador a indicar em cada vez o jogador a jogar. Em cada jogada é indicada a posição/casa em que o jogador fica após a jogada. No final escreve-se na consola o jogador ganhador e o número de vezes que cada jogador jogou. **(4 valores).**

```
import java.util.Scanner;

public class Main {

	public static void main(String[] args) {

		Game myGame = new Game(30);

		Scanner in = new Scanner(System.in);

		int player=0;

		while (                                          ){

			player=readInt(in);

			System.out.println("Jogador "+player+" ficou na casa
"+myGame.play(player)+ ".");

		}
```

```
		in.close();

	}

	private static int readInt(Scanner in) {

		int player=0;

		while (                                           ){

			System.out.print("Jogador (1 ou 2):");

			player = in.nextInt();

			in.nextLine();

		}

		return player;

	}

}
```

Nome: Número:

Grupo III

Considere o seguinte enunciado do problema:

Objectivo:

Representar um cartão de crédito e a sua utilização.

Descrição:

Um cartão de crédito tem um plafond, ou seja, um valor máximo que o seu proprietário pode gastar em cada mês. Este valor é fixado em 1500 EUR. Cada cartão tem ainda um custo anual fixado em 30.5 EUR. O proprietário pode efetuar compras de qualquer valor, mas nunca pode exceder o valor do plafond. No entanto, é também possível fazer pagamentos pontuais do valor em dívida, ou seja, é possível ir diminuindo o valor em dívida. No final de cada mês, deve ser possível pagar a totalidade do valor em débito.

Funcionalidades:

- Fazer um pagamento com o cartão de uma compra com um determinado montante (valor real). O método deve indicar se foi possível ou não efetuar a compra;
- Fazer um pagamento pontual de parte do valor em dívida;
- Verificar se é possível realizar uma determinada compra;
- Consultar o plafond ainda disponível;
- Consultar o valor em dívida;
- Pagar a totalidade em falta;
- Pagar o valor do custo anual com o próprio cartão;

A classe CreditCard especifica objectos que representam cartões de crédito. Indique a interface da classe CreditCard, e o método construtor.

Para cada método indique o tipo do seu resultado e o tipo dos seus parâmetros (se existirem). Explique ainda a sua finalidade (para que serve) de forma clara e intuitiva. Indique ainda que métodos são construtores, que métodos são modificadores, e que métodos são de consulta. (**5 valores**)

Nome: Número: