1. .

```
package sessão17_10;

public abstract class Evaluation {
	protected String description;

	/**
	 * Evaluation Constructor has the following parameter
	 * @param desc
	 */
	public Evaluation(String desc)
	{
		description = desc;
	}

	/**
	 * Presents the grade of the evaluation
	 * @return the grade
	 */
	public abstract int getGrade();


	/**
	 * Builds the String shown in console with the prefix and the description an the
grade of
	 * each evaluation object
	 * @param prefix
	 */
	public void list (String prefix){
		System.out.println( prefix + " " + toString());
	}

	/**
	 * Builds a String with the description an the grade of each evaluation
	 * object
	 */
	public String toString()
	{
		return description + " - " + getGrade();
	}
}
```

```
package sessão17_10;

public class Simple extends Evaluation{
	private int grade;
	/**
	 * Simple Constructor has the following parameter
	 * @param desc
	 * @param grade
	 */
	public Simple (String desc, int grade){
		super (desc);
		this.grade = grade;
	}

	public int getGrade(){
```

```
                return grade;
        }
}
```

2.

```
package sessão17_10;

public abstract class Complex extends Evaluation {
        protected Evaluation [] evalComponents;
        private int currentIdx=0;  // será utilizado para saber a posição do array que
está a ser ocupada
        /**
         * Complex Constructor has the following parameter
         * @param desc
         * @param maxComp
         */
        public Complex (String desc, int maxComp)
        {
                super (desc);
                evalComponents = new Evaluation[maxComp]; // criar um array onde fiquem
armazenados os dados referentes às notas das avaliações

        }

        /**
         * If possible, store the data for the different Evaluations in array
         * @param ev
         * @return true if the data was stored, and false if the array is full
         */
        public boolean add (Evaluation ev)  //armazenar os dados referentes às diferentes
avaliações
        {
                if ( currentIdx<evalComponents.length){ //apenas se o array ainda não
estiver cheio, insere o valor no array
                        evalComponents[currentIdx] = ev;
                        currentIdx++;
                        return true;
                }
                else
                {
                        return false;
                }
        }
        /**
         * Builds a String with the description an the grade of each evaluation object in
array
         */
        @Override
        public void list(String prefix){  // imprime a descrição e nota de todas as
avaliações inseridas
                for (int idx = 0; idx < evalComponents.length; idx++)
                {
                        evalComponents[idx].list("-> ");
                }
        }
}
```

```
package sessão17_10;

public class Average extends Complex{
	/**
	 * Average Constructor has the following parameter
	 * @param desc
	 * @param maxComp
	 */
	public Average(String desc, int maxComp) {
		super(desc  + " (Average) ", maxComp);
	}

	/**
	 * Determines the average of the grades entered
	 */
	@Override
	public int getGrade() {
		int auxMean = 0;
		int mean;
		for (int idx = 0; idx < evalComponents.length; idx++)
		{
			auxMean+= evalComponents[idx].getGrade(); //Obter a nota de cada
objecto armazenado no array
		}
		mean = auxMean / evalComponents.length;
		return mean;
	}
}
```