Ćwiczenie 1:

Aplikacja WWW umożliwia dodawania nowych studentów poprzez formularz. Klasa kontrolera i model zaprezentowany jest poniżej. Napisz test jednostkowy, sprawdzający poprawność działania kontrolera.

```
@Controller
public  class StudentController {

	private List<Student> students = new ArrayList<Student>();

	@RequestMapping(value="/students", method=RequestMethod.GET)
	public String student(Model model) {

		model.addAttribute("students", students);
		model.addAttribute("student", new Student());

		return "students";
	}

	@RequestMapping(value="/students", method=RequestMethod.POST)
	public String addStudent(@Valid Student student, BindingResult results) {

		if (results.hasErrors()) {
			return "students";
		}

		students.add(student);
		return "redirect:/students";
	}
}
```

```
public class Student {

	@NotNull
	@Length(min = 1, max = 50)
	private String firstname;

	@NotNull
	@Length(min = 1, max = 50)
	private String lastname;

	@NotNull
	@Email
	private String emailAddress;

	@NotNull
	@Past
	private Date dateOfBirth;

	@NotNull
	@Past
	private Date courseStart;

	@NotNull
	@Future
	private Date courseFinish;

	@NotNull
	@Pattern(regexp="^[A-Z]{3}\\-[0-9]{8}\/[0-9]{2}$")
	private String idNumber;

	//.. getters and setters
}
```

Instrukcja rozwiązania

1. Dołącz do projektu bibliotekę: spring-test
2. Utwórz test zawierający trzy metody: sprawdzającą pobieranie danych, błędny zapis oraz poprawny zapis.
3. Dla zapisu, symuluj pola formularza dodając parametry do obiektu HttpRequest
4. Dla poprawnego zapisu sprawdź czy wewnętrzna lista kontrolera zawiera odpowiedni zapis.

Rozwiązanie

```
@RunWith (SpringJUnit4ClassRunner.class)
@ContextConfiguration ({
	"file:src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml"
})
public  class StudentContollerTest {

	@Autowired
	ApplicationContext ctx;

	@Autowired
	StudentController controller;

	@Test
	public void servletTest() throws Exception {
		MockHttpServletRequest req = new MockHttpServletRequest(
			"GET", "/students");
		MockHttpServletResponse resp = new MockHttpServletResponse();
		HandlerAdapter handlerAdapter =
			ctx.getBean(AnnotationMethodHandlerAdapter.class);
		final ModelAndView model = handlerAdapter.handle(req, resp,
				controller);

		assertViewName(model, "students");
		assertAndReturnModelAttributeOfType(model, "student", Student.class);
		assertModelAttributeValue(model, "student", new Student());
		assertAndReturnModelAttributeOfType(model, "students", List.class);
		assertModelAttributeValue(model, "students", new
			ArrayList<Student>());
	}

	@Test
	public void expectingValidationErrors() throws Exception {
		Student s = new Student();
		s.setFirstname("John");
		s.setLastname("Smith");

		MockHttpServletRequest req = new MockHttpServletRequest(
			"POST", "/students");
		req.setParameter("firstname", s.getFirstname());
		req.setParameter("lastname", s.getLastname());

		MockHttpServletResponse resp = new MockHttpServletResponse();
		HandlerAdapter handlerAdapter =
			ctx.getBean(AnnotationMethodHandlerAdapter.class);
		final ModelAndView model = handlerAdapter.handle(req, resp,
				controller);


		assertViewName(model, "students");
		assertAndReturnModelAttributeOfType(model, "student", Student.class);
		assertModelAttributeValue(model, "student", s);

		BindingResult result = (BindingResult)
			model.getModel().get(
			"org.springframework.validation.BindingResult.student");
		assertTrue(result.hasErrors());
		assertFalse(result.hasFieldErrors("firstName"));
		assertFalse(result.hasFieldErrors("lastName"));
		assertTrue(result.hasFieldErrors("emailAddress"));
		assertTrue(result.hasFieldErrors("dateOfBirth"));
		assertTrue(result.hasFieldErrors("courseStart"));
		assertTrue(result.hasFieldErrors("idNumber"));
	}
```

```
	@Test
	public void savingStudent() throws Exception {
		SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy");

		Student s = new Student();
		s.setFirstname("John");
		s.setLastname("Smith");
		s.setEmailAddress("student@domain.com");
		s.setDateOfBirth(LocalDate.now().minusYears(20).toDate());
		s.setCourseStart(LocalDate.now().minusYears(5).toDate());
		s.setIdNumber("ABC-12345678/98");

		MockHttpServletRequest req = new MockHttpServletRequest(
			"POST", "/students");
		req.setParameter("firstname", s.getFirstname());
		req.setParameter("lastname", s.getLastname());
		req.setParameter("emailAddress", s.getEmailAddress());
		req.setParameter("dateOfBirth", formatter.format(s.getDateOfBirth()));
		req.setParameter("courseStart", formatter.format(s.getCourseStart()));
		req.setParameter("courseFinish", "");
		req.setParameter("idNumber", s.getIdNumber());

		MockHttpServletResponse resp = new MockHttpServletResponse();
		HandlerAdapter handlerAdapter =
			ctx.getBean(AnnotationMethodHandlerAdapter.class);
		final ModelAndView model = handlerAdapter.handle(req, resp,
				controller);


		assertViewName(model, "redirect:/students");
		assertAndReturnModelAttributeOfType(model, "student", Student.class);
		assertModelAttributeValue(model, "student", s);

		BindingResult result = (BindingResult)
			model.getModel().get(
			"org.springframework.validation.BindingResult.student");
		assertFalse(result.hasErrors());

		List<Student> students = (List<Student>)
			ReflectionTestUtils.getField(controller, "students");
		assertNotNull(students);
		assertEquals(1, students.size());
		assertEquals(Arrays.asList(s), students);
	}

}
```